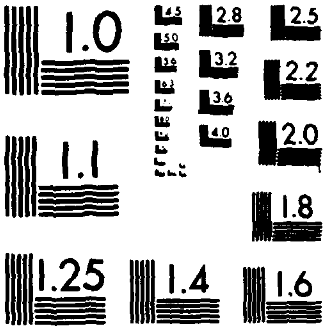
AD-A184 755 NETWORK RECONSTITUTION PROTOCOL(U) SRI INTERNATIONAL 1/1
MENLO PARK CA A E MATHIS ET AL JUN 87 SRI-5453
RADC-TR-87-38 F30602-83-C-0027

UNCLASSIFIED F/G 12/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

RADC-TR-87-38
Final Technical Report
June 1987

NETWORK RECONSTITUTION PROTOCOL

SRI International

Sponsored by
Defense Advanced Research Projects Agency (DOD)
ARPA Order No. 4715

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-87-38 has been reviewed and is approved for publication.

APPROVED:

CARL A. DeFRANCO, JR.
Project Engineer

APPROVED: Raymond P. Urtz

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: Richard W. Pouliot

RICHARD W. POULIOT
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTD) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

NETWORK RECONSTITUTION PROTOCOL

James E. Mathis
Barbara Denny
Robert Gilligan
Dwight Hare
Rodney Reining
Greg Skinner
Zaw-Sing Su

Contractor: SRI International
Contract Number: F30602-83-C-0027
Effective Date of Contract: 3 January 1983
Contract Expiration Date: 30 May 1987
Short Title of Work: Network Reconstitution Protocal
Program Code Number: 3T10
Period of Work Covered: September 1983 - May 1986

Principal Investigator: James E. Mathis
Phone: (415) 326-6200

RADC Project Engineer: Carl A. DeFranco Jr.
Phone: (315) 330-2805

Approved for public release; distribution unlimited

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Carl A. DeFranco Jr. (COTD) Griffiss AFB NY 13441-5700 under Contract F30602-83-C-0027.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD A184 [illegible]

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS
N/A

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release;
distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
SRI 5453

5. MONITORING ORGANIZATION REPORT NUMBER(S)
RADC-TR-87-38

6a. NAME OF PERFORMING ORGANIZATION
SRI International

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION
Rome Air Development Center (COTD)

6c. ADDRESS *(City, State, and ZIP Code)*
333 Ravenswood Avenue
Menlo Park CA 94025-3493

7b. ADDRESS *(City, State, and ZIP Code)*
Griffiss AFB NY 13441-5700

8a. NAME OF FUNDING/SPONSORING ORGANIZATION
Defense Advanced Research Projects Agency

8b. OFFICE SYMBOL *(If applicable)*
IPTO

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F30602-83-C-0027

8c. ADDRESS *(City, State, and ZIP Code)*
1400 Wilson Blvd
Arlington VA 22209

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.
62708E/ 63728F	D715	01	02

11. TITLE *(Include Security Classification)*
NETWORK RECONSTITUTION PROTOCOL

12. PERSONAL AUTHOR(S) James E. Mathis, Barbara Denny, Robert Gilligan, Dwight Hare,
Rodney Reining, Greg Skinner, Zaw-Sing Su

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM Sep 83 TO May 86

14. DATE OF REPORT *(Year, Month, Day)*
June 1987

15. PAGE COUNT
100

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
12	05	
23	05	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Communications Networks   Gateways   Dynamic
Protocol   Reconstitution   Routing
Packet Radio   ARPANET

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The network reconstitution protocol (RP) developed for the Strategic C^3 Experiment is based on the DARPA internetwork, taking the TCP/IP protocol suite as a point of departure.

In a dynamic operational environment, mobile platforms may cause networks to merge or to partition. A host or gateway may move its attachment from one network to another, and as hosts and gateways fails, new ones may be deployed. In order to accommodate these dynamics of internetwork topology, SRI has investigated an approach based on dynamic, gateway-centric addressing.

Each host affiliates itself with a local gateway. Gateways are indivisible entities during dynamic operations. Thus, the gateway-based internetwork hierarchy avoids resource-consuming reorganization which a network-based hierarchy must endure.

Address change is unavoidable for dynamic operations. A host losing connectivity with its affiliated gateway would change its affiliation to a different gateway, thus changing (over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED □ SAME AS RPT. □ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Carl A. DeFranco, Jr.

22b. TELEPHONE *(Include Area Code)*
(315) 330-2805

22c. OFFICE SYMBOL
RADC (COTD)

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Block 19. Abstract (Cont'd)

its address. For efficient operations, a 32-bit "address" is used in the DARPA Internetwork for both identification and addressing. To maintain an end-to-end connection during a change of address, a semantic separation of the host identification (its name) from its address would be necessary. A mechanism for dynamic separation of semantics is incorporated into the RP design. After separation, the original address becomes a name (or TCP connection identifier). Dynamic separation of semantics accommodates network dynamics while maintaining efficiency during quiescent periods.

Changing address in the midst of a connection would require packet forwarding. A "forward request protocol" installs forwarding information in selected gateways. This distributed forwarding database facilitates the logical name to physical address binding.

A variation of the Ford algorithm was adopted for routing. For quick response to a local topological change, the incremental nature of a Ford algorithm calls for limited routing updates to resume traffic. The proposed RP architecture will work, however, with a variety of routing schemes, including shortest-path-first (SPF).

Backward compatibility in addressing format allows the RP hosts and gateways to operate as an independent autonomous system within the DARPA internetwork.

Contents

1 Introductiɔn 1

2 Goals of Reconstitution Experiment 3

3 Background 6
3.1 Dynamic Address/Routing Hierarchy 8
3.2 Name/Address Decoupling 9
3.3 Limited Logical Addressing 10

4 Reconstitution Protocol Architecture 12
4.1 Finding Gateways 13
4.2 Host Addresses 14
4.3 Default Gateway Association 14
4.4 Address-Identifier Mapping 15
4.5 Requesting Forwarding 15
4.6 Forwarding of Logically Addressed Packets 16
4.7 Routing 16
4.8 Gateway Clustering 17

5 Implementation 18
5.1 RP Gateway Implementation 19
5.1.1 Host Affiliation 21
5.1.2 Neighbor Gateway Acquisition 22
5.1.3 Routing 23
5.1.4 Forward Request Protocol 27
5.1.5 Exterior Gateway Protocol 31
5.2 Terminal Interface Unit Modifications 31
5.2.1 TELNET 32
5.2.2 TCP 32
5.2.3 IP 32
5.2.4 ICMP 32
5.3 VAX/Sun Unix 33
5.3.1 Interface Drivers 34

5.3.2 Routing 35
5.3.3 IP Packet Handling 35
5.3.4 User-Level Code 36

6 Reconstitution Experiments 37
6.1 PRNET Partitioning and Merging–F5/10 37
6.2 ARPANET and Multiple Network Partitioning–F8/11 41
6.3 PRNET Network Mobile Host–F9 44

7 Conclusions and Observations 49
7.1 Functionality 49
7.2 Performance 50
7.3 Overall Architecture 51
7.4 Ease of Implementation 51

A Host-Gateway Protocol 53
A.1 General Interaction Model 53
A.2 Host FSM Variables 55
A.3 Host Finite State Machine 55
A.4 Gateway Functions 57
A.5 Host Traffic Management 57
A.6 Network-Specific Details 58
A.6.1 PRNET Details 58
A.6.2 ARPANET Details 58
A.7 Unique Host ID Assignment 59
A.8 New IP Options 59
A.8.1 Old-Source-Address Option 60
A.8.2 Old-Destination-Address Option 60
A.9 Host-Gateway Packet Formats 60

B IPLI Compatibility Considerations 69
B.1 Address Format 70
B.2 Unique Host Identifiers 70
B.3 Gateway Acquisition Protocol 71
B.3.1 Finding Operational Gateways 71
B.3.2 Monitoring Host-Gateway Status 72
B.3.3 Getting the New Associated Gateway Number 72
B.3.4 Specifying Previous Addresses 72
B.4 Handling of Old-Destination-Address IP Option 73
B.5 Handling of Old-Source-Address IP Option 73
B.6 More Sophisticated Fault-Isolation Mechanisms 73
B.7 Initialisation 74
B.8 Summary of Recommendations 75

C Gateway-Gateway Protocol **76**

List of Figures

5.1 Gateway Process Structure 20
5.2 Example Internetwork Topology 24
5.3 Distance Matrix for Gateway B 25
5.4 Routing Matrix for Gateway B 26
5.5 New Distance Matrix for Gateway B 28
5.6 New Routing Matrix for Gateway B 29

6.1 F5/10 Demonstration—Coalesced PRNETs 39
6.2 F5/10 Demonstration—Partitioned PRNETs 40
6.3 F8 Demonstration–Initial Configuration 42
6.4 F8 Demonstration–Partitioned ARPANET 43
6.5 F11 Demonstration–Partitioned ARPANET 45
6.6 F11 Demonstration–Partitioned ARPANET and PRNET 46
6.7 F9 Demonstration—Network Mobile Host 47

Chapter 1

Introduction

The potential of achieving resource-sharing by interconnecting various types of packet-switched networks led to the development of internetworking. The current proliferation of packet-switching networks, local area networks, and gateways has resulted in a large internetwork system with rich connectivity and hence potential survivability in the face of node and link failures. Unfortunately, the current DoD internetwork architecture and protocols[1] were developed with a static environment in mind; that is, one in which hosts do not quickly relocate and networks do not quickly merge or partition. The current internetwork is concerned with topology changes bought about primarily by failures in gateways rather than substantial changes in network structure or organisation; thus the potential for survivability of the internetwork system is largely unrealized.

When packet-switching and internetwork technologies are applied to the real-world problems of the military services, deficiencies often become apparent[2,3]. In the Strategic $\mathrm{C}^3_{\mathrm{u}}$ Experiment packet-radio technology[4] offers flexibility of operation and support for mobile nodes (aircraft); however, this flexibility and mobility is not adequately supported at the internetwork level. Early in the experiments three primary deficiencies were uncovered that relate to:

- Partitioning of networks
- Merging of networks
- Maintaining communications to network-mobile hosts.

In all of these cases, fundamental (but perhaps implied) assumptions of the internetwork architecture were being violated.

These problems are an outgrowth of the development of new network technologies over the past 10 years. Prior to the development and proliferation of computer networks utilising radio transmission from easily movable nodes, the problems of merging and network-mobile host did not exist: land-line-based networks are not often reorganized. While partitioning has always been a potential problem, the desired degree of survivability was always achieved through redundancy (additional links and nodes) internal to the network.

The following report covers work performed by SRI International from September 1983 through May 1986 on design and implementation of, and experimentation with, enhanced internetwork reconstitution protocols.

Chapter 2

Goals of Reconstitution Experiment

The Strategic C^3 Experiment was conceived to develop and exercise advanced computer processing and communication technology with a demonstration application of aircraft recovery. Because an eventual operational system would operate in a post-attack environment, issues of survivability, reconstitutibility, and mobility were important in guiding the technological improvements. Advances in packet-switched radio systems provide a high-speed digital communication system that is self-organizing, automatically directed, usable on mobile nodes, and suitable for experimental use on aircraft as well as land-mobile and fixed-base stations.

While the PRNET technology for network management solved many of the technical issues for self-directing networks with mobile nodes, and the Survivable Radio Network (SURAN) program concentrated on improvements in survivability and network size, it was recognized that a single communications technology or a single network cannot fully support all of the diverse requirements of a strategic C^3 system. The issues of survival and reconstitutable communications must be addressed from an internetwork perspective that can be used with a variety of network technologies, including land-line, local area, satellite and broadcast and point-point radio. However, internetwork technology lags in terms of its ability to reconstitute after failures.

Thus, the goals of the reconstitution protocol effort are two-fold: to develop a system that can demonstrate the utility of packet-switching within the context of the Strategic C^3 Experiment and to gain experience in the problems associated with a dynamic internetwork system so that future efforts can build upon this work.

The technical challenge was to design and implement extensions to the existing internetwork protocols so that they quickly and automatically adapt to major changes in the underlying internetwork topology in a manner that is transparent to applications-level protocols and is accomplished with minimum disruption to data transport. This challenge was subject to the following practical constraints:

- The protocol must maintain existing TCP connection across disruptions; it is not acceptable to require closing/reopening of TCP connection.
- The protocol must be compatible with existing IP system as much as possible. It must be possible to:
  - Communicate with unmodified TCP/IP hosts on the ARPANET.
  - Exchange network reachability information with normal IP gateways.
- There must be no changes to the TCP protocol.
- There should be little or no change to the Internetwork Private-Line Interface (IPLI) communications security device.

A series of demonstrations was defined that exercised selected capabilities of the PRNET and internetwork communications system required to support strategic C^3 systems of the type of interest to SAC. The demonstrations highlighted both existing capabilities of the PRNET and internetwork to reconfigure, and desired capabilities for network reconstitution. For the latter functions, the demonstrations also served to focus attention on the problems and provide a framework for measuring the effectiveness of solutions. Using the nomenclature of the Experiment Program Plan[5], the demonstrations involving network reconstitution were:

- F5–Partitioned PRNET
- F8–Partitioned ARPANET
- F9–Mobile Airborne PRNET Host
- F10–Coalescing PRNETs
- F11–Multiple Partitioned Networks (ARPANET and PRNETs).

For purposes of efficiently using SAC aircraft for the demonstration and testing, we collapsed this list into three experiments by combining related functions.

Thus the partitioned PRNET and coalescing PRNETs were combined, since partitioning and coalescing are opposite sides of a cycle. Similarly, partitioned ARPANET and multiple partitioned networks were combined. The resulting experiment list, ordered by increasing protocol functionality or complexity, is:

- F5/10–Partitioning and Coalescing of PRNETs
- F8/11–ARPANET and multiple network partitions
- F9–Network Mobile Host.

Because the reconstitution protocol (RP) implementation in both the gateways and hosts was evolving, the experiments and demonstrations were conducted in the order of least to most complex. This ordering allowed us to experiment, test, and verify basic RP functions and then to build upon these functions in later experiments. The details of the experiments are described in Chapter 6.

To accomplish the long-range goal of understanding the problems, we developed a general architecture for handling dynamics in internetwork topology that can be applied to these particular experiments. The technical approach is discussed below, followed by the experiments conducted and the conclusions reached.

Chapter 3

Background

Before discussing the details of RP, it is useful to review some of the underlying framework upon which it was developed.

The DARPA internetwork is essentially a static, two-level hierarchy with networks of hosts interconnected by gateways; this hierarchy is tightly bound to the addressing scheme of *net.host*, where *host* identifies the subscriber on network *net*. Conforming to the address structure, the routing also follows a two-level hierarchy in that route computations are first performed to locate the network, then each network is responsible for delivery to the destination host.

Although the routing is not strictly required to conform to the addressing hierarchy, the internetwork systems components (gateways, hosts, and networks) lack protocols (hence information) to route other than along the address structure. Some protocols and experiments have been proposed to decouple the routing from addressing to solve specific problems, such as expressway networks, mobile hosts, or partitioned networks[6,7]; however these protocols have generally not been implemented.

The routing structure is thus a static hierarchy since it is identical to the static address structure, with its human-assigned network and host numbers. Thus the deficiencies in the routing are derived from a tight coupling of routing to addressing (for efficiency) and a static address structure (for simplicity).

A static hierarchy does not mean that the internetwork system cannot invisibly handle link and nodes failures, because failures do not always change the hierarchical structure. But changes in the topological hierarchy, such as by merging of networks, must be reflected by changes in addresses; these changes are not automatic and are visible to higher protocol layers. For example, the failure of one of the ARPANET-MILNET gateways is automatically handled by rerouting traffic through other ARPANET-MILNET gateways. However, the split of the original ARPANET into the MILNET and residual ARPANET (a very large network partitioning) happened smoothly only because of substantial planning and preparations for the required address changes.[1]

Unfortunately, the reconstitution of strategic networks must cope with unexpected failures that can doom simple preplanned address change procedures. The three deficiencies that were of main concern in the RP effort are the direct result of the consequences of disrupting the static hierarchy.

The RP system extends the internetwork architecture to keep the two-level hierarchy that binds addressing and routing for efficiency, but adds a dynamic addressing hierarchy. Since routing is easiest when performed according to some regular structure, the addresses are allowed to change as the overall topology of the internetwork changes or as hosts move from network to network to reflect current routing information.

With the addressing/routing hierarchy dynamic, the gateways can route traffic in a flexible manner to handle network reconstitution, but the dynamics are still not invisible. A further assumption at the transport protocol level (TCP) is that the connection end-point identifier at the transport level is identical to the address at the IP level. Thus any changes to the IP address, as a result of movement to a new network, for example, disrupt all existing transport-level connections to the host whose address has changed.

To provide invariant indicators for the transport level (required by TCP) but flexibility at the IP address level, dynamic binding is introduced within hosts and gateways so that the TCP identifier (name) stays constant while the IP address changes to reflect changes in topology.

As described, the main technical approach to the network reconstitution protocol centered around:

- Replacing the static two-level addressing/routing hierarchy with a dynamic hierarchy.
- Adding a dynamic binding between IP address and TCP connection identifier.
- Introducing a limited form of internetwork logical addressing.

[1] The planned addressing changes were a small part of the substantial effort required to plan and execute the trunking changes and node rehomings that constituted the actual network separation.

These three key features are the basis of the RP work and were chosen for both backward-compatibility with existing TCP/IP implementations and efficient use of network resources.

3.1 Dynamic Address/Routing Hierarchy

An obvious approach to handling the problems brought about by changes in network organization is to eliminate completely the concept of network and have a flat addressing structure. Since the concept of network would no longer exist, handling of network partitions, network merging, and host movement between networks becomes moot. Unfortunately, the actual problem of how to route user traffic does not disappear; rather the amount of overhead traffic to exchange routing information increases dramatically and becomes unmanageable for large networks.

Current routing algorithms exchange an amount of information at least proportional to the number of nodes; introducing a flat address space would increase the number of nodes involved in the route computations and information exchanges from a few hundred to several tens-of-thousands. Thus efficiency considerations dictate some form of hierarchical routing.

In homogeneous networks, such as the SURAN,[2] dynamic clustering (perhaps based on elections of cluster heads) is feasible. However, in the internetwork, gateway nodes have functionality (in terms of network connectivity) much different from host nodes; thus the hierarchy must ouviously revolve around gateways. A clustering approach based instead on the individual packet-switches would not be practical because of the substantial modifications required in all packet-switches.[3]

Thus the RP architecture is organised in a gateway-centric manner rather than in a network-centric manner. The gateway-centered approach immediately brings several advantages, the primary one being that gateways, as physical entities, do not partition; the gateway's interface to the network may fail, but this is a boundary condition at a network partition. Networks, on the other hand, being artificial entities are subject to a variety of vagaries such as partitions and mergers.

A further refinement of the model is to route based on gateway-halves; a

[2] A new radio network technology to handle increased network size and survivability.

[3] The construction of a network comprising a variety of different transmission media is a separate topic of research. Here we have constrained ourselves to considering the internetworking approach to heterogeneous transmission systems.

gateway "half" being associated with each network interface. While we use the term gateway-half, the approach is trivially extended to cover gateways of arbitrary number of network connections (e.g., three-headed gateways), and we use the term in this generic manner.

Routing based on gateway halves, or interfaces, has a number of advantages, including solving the triangle routing problem, reducing the counting to infinitity problem in Ford-type routing algorithms, solving the ambiguity of how to address a gateway, and providing a unique way to identify groupings of hosts (as discussed below).

3.2 Name/Address Decoupling

The second primary deficiency in the DARPA internetwork is the close coupling of IP address (for routing purposes) and TCP address (for connection identification purposes). In the original design of TCP, the use of the IP address was perfectly reasonable. IP addresses rarely changed—large mainframes seldom changed IMP ports on the ARPANET, the PRNET used logical addressing within the network to handle intranetwork mobility, and network number assignment was permanent. Also, the overhead of TCP/IP was considered high enough at the time without introducing an additional 64 bits of TCP source/destination connection identifier.

Thus, the TCP/IP addressing scheme cleverly collapsed the three elements of routing, addressing, and identifying (how, where, and what) into a single 32-bit field. Unfortunately, since connection identifiers must be invariant for the duration of a connection, the addressing and routing information must also remain static for the duration of a connection. Thus any changes in address or routing indicators necessarily disrupt established transport connections.

The only solutions are to allow reestablishment of transport connections (perhaps under control of a session protocol) or to allow the IP address to change. Since reestablishment of transport connections would involve the loss of data (unless the session protocol had full error control duplicating the TCP functions!), the ability to change the IP addresses while maintaining TCP identifiers was included in the design of RP.

To maintain backward compatibility and allow for interoperation with normal TCP/IP hosts, the address changing is handled through an IP option. Specifically, at the source host's IP layer, old source and destination addresses are copied into an OLD-SOURCE-ADDRESS or OLD-DESTINATION-ADDRESS option, respectively, and the current IP address placed in the IP header. The current destination address thus controls the routing of packets to the destination, and the current source address is available for the return of ICMP messages. At the destination host's IP, the old addresses are copied back into the IP header (or equivalent data structure) before passing the packet to TCP or other higher-level protocols.

Another way to view these option is as TRANSPORT PROTOCOL SOURCE IDENTIFIER and TRANSPORT PROTOCOL DESTINATION IDENTIFIER options. If the identifier option is not present (the normal case), the identifier is taken to be equal to the IP address field. If the option is present, its value is used for the connection identifier.

The options, however, are implemented at the IP level rather than TCP for two reasons. First, the ability to change addresses while maintaining a transport connection may be of value to protocols other than TCP. Secondly, the gateway may be required to insert these options in the process of forwarding a packet to a host affiliated with a different gateway; to preserve layering, the option must thus be at the IP level.

3.3 Limited Logical Addressing

Using a dynamic addressing hierarchy allows the gateway system to route around partitions and to handle merged networks by changing the address of hosts. The provision for separation of IP address from TCP connection identifier allows a host to change its address while maintaining TCP connections intact. The missing link is a mechanism that quickly distributes the information about host address changes so that a transmitting host can correctly change the destination's IP address to track changes in network topology or host movement. The operation of this mechanism is further complicated by unpredictable addresses changes and by both hosts changing addresses simultaneously.

The RP architecture utilizes a two-level mechanism. For long response times, the domain name service must be capable of changing the address of host names registered in the database; a host would be responsible for updating its name→address translation entry using a protocol beyond the scope of this effort. The old address should be purged from the name system within 30 minutes of an address change. However, updating an address in the name server does not solve the problem for existing connection or connections established before the name database is updated (which may take many minutes in a large domain system with several servers).

To solve the short-term problems, a limited form of internetwork logical addressing is used. In this type of logical addressing, the old address for a host is dynamically rebound to redirect packets to the new host address. The service is limited in that it is not intended to serve as a general logical address mechanism that can be used to locate resources, but as a temporary "bridge" until the new host address is propagated and the use of the old address dies away.

When considering a logical addressing mechanism, it is important to consider the replication of logical-physical address mapping knowledge, how the database is maintained, and where the logical-to-physical address translation is actually performed.

In the ARPANET and PRNET logical addressing, the logical-physical address mapping information is broadcast to every switching node in the network; usually this information is carried along with the route updates. The transformation to physical address is done in the node where the packet enters the network and the packet is then routed directly to the destination using the physical address. With the entire mapping database distributed to every node, this scheme is robust against node failures.

Other approaches concentrate the mapping function in server nodes. The entry node must recognise and route the packet to the logical mapping server, which then translates the address. The servers must replicate the mapping database for reliability; even so, there is no way to guarantee that every partition will contain a server.

The RP architecture adopts a compromise position. Because of the large numbers of gateways and hosts in the networks, global distribution of the logical-physical mapping information is infeasible. We instead concentrate the mapping function in the gateways "close" to the old location of the host. That is, the packet is routed towards the (now old) address; at a gateway close to the old destination address, the current address is put into the packet which is then forwarded on. This approach provides for a natural distribution of the forwarding load, concentrates logical mapping information, and provides some level of redundancy. The details are described below.

Chapter 4

Reconstitution Protocol Architecture

As indicated earlier, the basis of our approach was three-fold: first, to recognize that the same field should not be both an address to the routing layer (e.g., IP) and a host-name identifier at the transport level (e.g., TCP); second, to restructure the internetwork organisation around gateways rather than around networks; and third, to introduce a limited form of internetwork logical addressing.

The first premise has been independently adopted by most designers of new network and transport protocols and is a part of the emerging ISO internetwork architecture. By performing a dynamic binding between *connection identifier* and *host address*, we have a mechanism that will allow changes in the address to reflect changes in the internetwork topology (for efficient routing) while keeping an invariant name at the transport level to identify the connection.

The approach of using a gateway-centric organisation is not so widely adopted because of the feeling that "gateways connect networks" rather than "networks connect gateways." However, switching to a gateway-centric structure makes it easier to track changes in internetwork topology, since gateways can be considered as indivisible entities that do not partition or merge.

While our approach only transforms the problems of network reconstitution into those of maintaining the identifier-address mapping and the fabric of gateway routing and organization, we believe that these problems can be solved in a general manner, rather than by engineering specific solutions to a variety of network reconstitution and internetwork topology issues that arise in the existing system. These problems areas can be further divided into:

- Discovering gateways (by hosts and other gateways)
- Picking the correct address for a host
- Maintaining the address-identifier mapping
- Operating if no gateway is found
- Requesting forwarding of packets sent to old addresses
- Forwarding of logically addressed packets
- Routing among the gateways
- Organizing gateways into clusters.

Additional discussion on the background of the architecture can be found in [8,9,10].

4.1 Finding Gateways

A basic practical problem of the existing internetwork implementation is that the addresses of gateways are maintained in static tables residing in hosts and gateways. Clearly, if networks can merge and hosts move between networks, the hosts and gateways must have the ability to discover gateways in a dynamic manner. The problem of gateway discovery can be broken down into two cases:

- Discovering the "closest" gateway on the network
- Discovering all gateways on a network.

The first procedure is required by a host before it can start the affiliation process. While the discovery process may yield multiple gateways, and those choices may not be optimum, it is required only to supply a single gateway. However, if the internetwork system is to work well and not self-partition, each gateway must be able to discover every other gateway attached to the local network. Unfortunately, details of the discovery process are network specific and will likely be different for hosts than for gateways. The specific process is described for ARPANET and PRNET in Appendix A.

4.2 Host Addresses

As described earlier, the RP is organised around a gateway-centric model to support the network reconfiguration dynamics while maintaining efficiency. Rather than having a fixed, assigned network number as part of its address, the host's address contains a field that identifies the gateway "closest" to the host. For interoperability with the existing DOD internetwork hosts and gateways, the gateway ID is allocated 14 bits and the host ID is also allocated 14 bits so that gateways can be addressed as hosts in a similar manner.[1] The resulting 32-bit IP address is thus:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|   Unique Gateway ID       |0 0|   Unique Host ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

To the current internetwork system, this 32-bit address appears to be a Class-B address with the host attached to the Class-B network identified by the gateway ID. This address definition is compact and is compatible with existing internetwork addresses, and allows interoperations of the reconstitution gateways with the current internetwork system via the Exterior Gateway Protocol (EGP).

The host ID field of the address is a unique number assigned to each host in the RP experiment; this ID is invariant across relocations of the host to new networks. The gateway ID is picked by means of a protocol between the host and a nearby gateway. Essentially, the host affiliates itself with a gateway and then adopts the gateway's ID as part of its address. The details of the affiliation process are described in Appendix A.

4.3 Default Gateway Association

If the requirement that a host associate with a gateway is taken strictly, the gateways form a common point-of-failure, in that hosts on a partition of a network without a gateway cannot communicate among themselves. To eliminate

[1] While this bit allocation is inadequate for a large internetwork system with many networks, gateways, and hosts, it is sufficient to conduct our experiments in network reconstitution, and increasing the size of the fields has no impact on the operation of the protocol.

this problem, the concept of default gateway association is used. If a host cannot contact a gateway, it constructs its address using zero (0) for the unique gateway ID and uses this address as it would any valid address, including registering it with the domain name service.

4.4 Address-Identifier Mapping

To support mobile hosts that can move from network to network while maintaining data connections, a separation is needed between the connection identifier used in TCP and the address used in IP. The TCP identifier would thus be invariant during the life of the connection, but the IP address would always correspond to the current "location" of the host in the internetwork. The IP address could thus always be used to make routing decisions on how to reach the destination host. Unfortunately, the design of the TCP and IP protocols has both protocols sharing the same pair of 32-bit address fields for source and destination identifiers.

While there are various ways to handle the double-addressing, we selected an approach that is generally backward-compatible with existing IP implementations. The address fields in the IP header always indicate the current IP address of the source and destination hosts and change as hosts move from network to network. If the IP address is different from the TCP identifier, an option is added to the IP header to carry the TCP identifier. The formats of these options are defined in Appendix A.

4.5 Requesting Forwarding

When a host changes its address (following relocation, gateway failure, or network partition) it can request forwarding of packets sent to its old address; this mechanism is similar to mail forwarding. The host sends a FORWARD-REQUEST packet to its affiliated gateway containing a list of its prior addresses that are still in use. An address is consider *in use* if it is still registered in a name server or is used by an existing TCP connection. The affiliated gateway then duplicates this information and forwards it to the gateways with which the host was previously affiliated. These affiliated gateways then return FORWARD-REPLY packets as positive acknowledgments.

4.6 Forwarding of Logically Addressed Packets

The final mechanism to support host dynamics is the forwarding of packets sent to the previous address of a host to its current address; we consider these packet as being logically addressed, in that the address no longer identifies the physical location.

Two mechanism exist to handle logically addressed packets:

- Gateways will readdress and forward packets according to the forwarding table maintained in each gateway.
- An ICMP control message is sent to the source to tell it to rebind the connection to eliminate the logical address.

When forwarding the packet, the destination address from the header is copied into an IP option (see above), and the host's current address is copied into the packet. An ICMP READDRESS message is then sent to the source host. Upon receipt of the READDRESS message, the source host should rebind the connection so that further packets are sent directly to the host at its new address, with the old destination address carried as an IP option. Thus the logical addressing forwarding load on the gateways is keep to a manageable level.

4.7 Routing

Although it is not strictly a part of the RP architecture, the performance of the routing algorithm is certainly important. For these experiments we have selected a variation of the Ford algorithm. The incremental nature of the Ford algorithm provides for limited routing updates to resume traffic at the penalty of "counting to infinity" when a gateway becomes unreachable. The RP architecture will work with other types of route computation algorithms, such as shortest-path-first (SPF).

The number of gateway-hops (as opposed to network hops) is used as the routing distance metric. Taking advantage of zero being the additive identity, this distance metric provides an implicit delineation of network boundaries. Two internetwork components attached to the same network are zero distance apart. A destination host can thus be reached through any local gateway in addition to its affiliated gateway.

For routing metrics other than gateway-hops—such as delay, marginal bandwidth or other types of service-related measures—the route computations would need to be separated from topology management. It is important to allow traffic entering a network to be sent directly to the destination host rather than to force the traffic through the affiliated gateway. The sero-distance measure is also used to define pseudo-networks or gateway clusters; that is, the collection of hosts and gateways that are directly reachable across a given network. The use of gateway clusters is described below.

4.8 Gateway Clustering

The reconstitution scheme presented so far works well for handling partitions, merges, and network-movement as long as there are no gateway failures. Clearly, mechanisms must be introduced to handle failed gateways. When a gateway fails, it:

- Can no longer pass traffic
- Loses the host's forward-request database
- Cannot forward logically addressed packets.

As long as the network is not isolated and an alternate path is available, the critical loss when a gateway crashes is in the forwarding database and in the ability to forward logically addressed packets. To survive crashes, these databases and functions are replicated to every gateway in a gateway cluster. That is, cross-net gateway neighbors also exchange forwarding databases so that any gateway in the cluster, not just the gateway that the host was affiliated with, can forward traffic to a host that has relocated.

But to be forwarded, traffic must still be delivered to the gateway cluster. The "mark" in the route computation provides this. Essentially, upon detecting a failure in a cross-net neighbor, each gateway in the cluster reports a distance of $0'$. The marking is propagated along with the distance information. In terms of distance comparisons, any noninfinite-distance route is preferred over any route with a marked distance. Thus, if the gateway has truly failed and not simply moved, the gateway cluster will continue to announce a phantom route to the failed gateway as long as entries referring to the failed gateway's ID are in the forwarding database.

Chapter 5

Implementation

While theoretical work on communication architectures is certainly the first step, the task of designing new communication protocols is not complete until tests of prototype implementations are conducted. These tests are used to uncover unanticipated problems and to gain a further understanding of the system while operating in a realistic environment.

In the case of the RP work, we clearly needed to prototype an RP gateway. To test the protocols in different host environments and to satisfy a variety of pragmatic constraints, we selected host implementations in the terminal interface unit and Unix workstations used in the Strategic C^3 Experiment.

The design of RP, while changing some semantics such as the meaning of an address, required few modifications to host IP software. Most of the mechanisms are already implemented in most hosts, such as gateway echoing, fault-isolation, and host-specific redirects. As an indication of the effort at host implementation, we describe two typical implementations: the terminal interface unit (TIU) and VAX-Unix running the BBN TCP/IP kernel. Overall, because of differences in internal structures and design philosophies, the effort to modify the Unix kernel was more than an order-of-magnitude greater than the effort required for the TIU; this effort was mainly directed toward defeating the complicated routing mechanisms built into Unix.

5.1 RP Gateway Implementation

To demonstrate the new design and test the protocol, we developed a gateway to interconnect ARPANET and PRNETs. This development emphasized only the routing and RP aspects of the gateway; the issues of remote maintenance, debugging, and monitoring were not significantly explored because they do not differ significantly from the non-RP case.[1]

The RP gateway is built upon the MOS operating system and runs on the existing LSI-11 gateway hardware used in the SAC testbed. The limitations of this hardware in terms of address space and memory posed problems for gateway development and precluded the integration of all of the software developed into the gateway.

To reduce the implementation effort, much of the packet-handling software was derived from the code implemented in the TIU: in particular, the network handlers for ARPANET and PRNET and much of the IP packet-handling code.

The RP implementation separates the gateway into two (or more) gateway halves following the gateway-half-centric approach used in RP; this separation was followed in the design of the process structure of the gateway. The gateway-halves coexisting in the same gateway are known as common-gateway (CG) neighbors, while gateway-halves that *interface* to the same network are known as cross-net (XN) neighbors. It is important to keep in mind the distinction of gateway-halves, since this concept is used extensively in the routing process.

The process and modul: structure is shown in Figure 5.1. Each gateway-half has its own local net process which handles the interface to the network. To reduce the number of context switches, IP/ICMP protocol handling was not implemented as a separate process; rather received IP packets are processed by library routines called from the local net dispatcher. If the packet is an RP packet (IP protocol 9), it is then handed-off to the RP handler process associated with that gateway-half.

All the information regarding each gateway-half is kept in a structure called *rp_struct*. The most important fields in this structure deal with the neighbor, distance, routing, and forwarding tables which will be discussed below.

The GWYCON process provides the user interface to the gateway. An operator can ask for the display of specific data structures resident in the RP handlers and the local net dispatchers. A packet printer is also available, with the user having the capability to selectively filter or print packets.

[1] An operational system should probably include automated fault detection and diagnostic systems, perhaps based on an expert systems approach, to assist in the repairing of communication assets. These issues were beyond the scope of this effort.

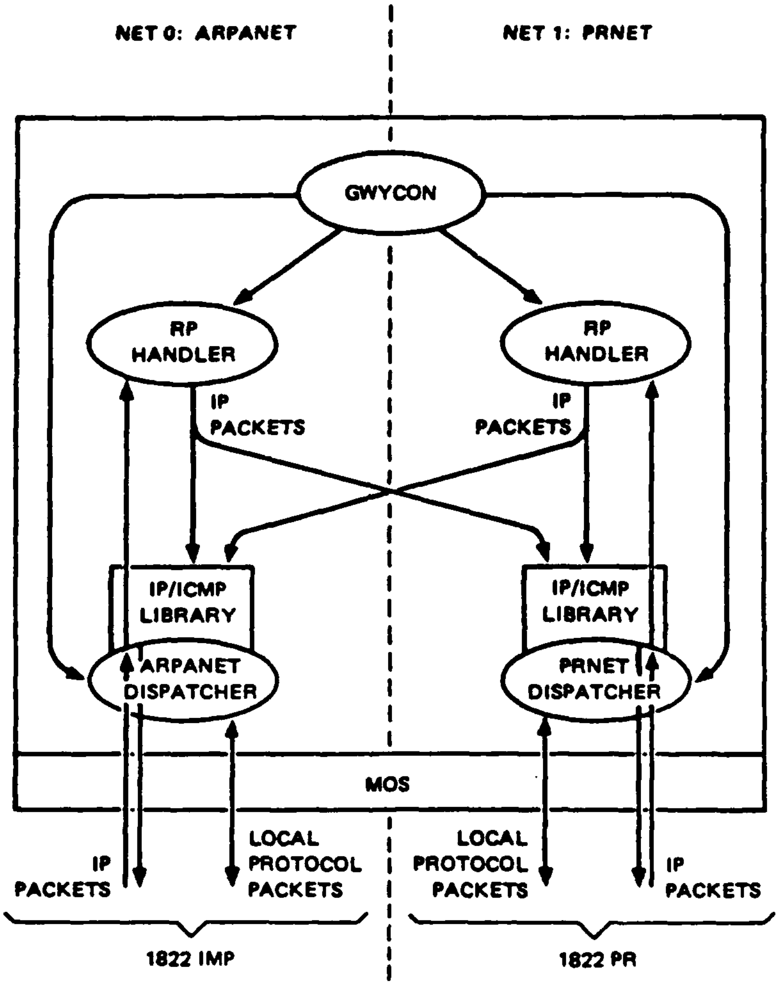


Figure 5.1: Gateway Process Structure

Rather than provide a detailed description of the software on a module basis, we present below a discussion of the key functions of RP implemented in the gateway:

- Host affiliation
- Neighbor gateway affiliation
- Routing
- Forward request handling
- Exterior gateway protocol.

5.1.1 Host Affiliation

When a host initializes or first becomes attached to a network, it must affiliate with a RP gateway as discussed earlier. By this process the host adopts an appropriate IP address that is based on its local network identifier concatenated with a gateway address supplied by the affiliating gateway.

The host is the active party in establishing and maintaining the affiliation, while the gateway is very passive and does not even maintain a table of affiliated hosts. In acquiring the affiliation with a gateway, a host sends an affiliation request (also known as an ICMP ECHO packet, type 8) to a gateway, and the receiving gateway returns an affiliation reply (ECHO-REPLY packet, type 0).[2] The host then chooses a gateway with whom to affiliate based on those gateway-halves from which it has received replies. Without benefit of other information, the host can choose the gateway that replied first. The host continues to monitor its affiliated gateway to verify the gateway's continued operation.

Because the PRNET does not provide a reliable indicator of nondelivery, the PRNET hosts must monitor its gateway by "pinging" at a low rate. To "ping," a device such as a host or gateway sends a special packet (usually ECHO packet) to another device at regular intervals and expects to receive a reply or acknowledgment for every packet of this type sent. It is through this mechanism that a device hopes to determine whether another device is up. However, if a gateway misses a certain number of pings, the host will try to affiliate with a new gateway-half. Various optimizations are possible to reduce

[2] Initially, the affiliation process was conducted using new packet types defined for ICMP. Because of prototype implementation constraints discussed in Appendix B, the semantics of affiliation were overloaded onto the ECHO and ECHO-REPLY ICMP packets. This dual usage was a simple and generally harmless expediency and is not recommended for future designs.

the rate of pinging, including use of network-specific status information and noting successful transit of user information through the gateway.

The gateway maintains no knowledge of which hosts are affiliated with it, so no resources are required on the gateway's behalf except for the ability to respond to an affiliation request (ECHO packet).

5.1.2 Neighbor Gateway Acquisition

Not only must hosts "discover" gateways, but gateways must have a mechanism to dynamically discover the existence of other gateways on the local network; otherwise, the system could not operate efficiently following a network merger.

Since there is no universal generic addressing scheme for each type of network in the internetwork, cross-net neighbor gateways learn about each other's existence through different techniques. For the present, the ARPANET gateways have an internal table of possible PSN/port pairs to search; in the future, ARPANET logical addressing could be used. For the PRNET, the gateway scans the LROP (local repeater on packet) received periodically from the radio for possible gateway addresses.

A neighbor can be in three possible states: *UP*, *DOWN*, and *ECHOING*. Once a neighbor is detected, the status of this new neighbor is *ECHOING*. If it answers a ping, it is marked as *UP* to quickly establish new connectivity. If the gateway misses two echo packets in a row, it returns to the *ECHOING* state. After missing 4 echo packets, it is considered *DOWN*. The first echo packet it receives always brings the status back to the *UP* state.

All these parameters can be adjusted for better operational efficiency or faster response times. Currently, the pinging interval is set to once every ten seconds. If the gateway being pinged misses more than four echo packets, this pinging interval increases to once every thirty seconds. If a gateway is brought back to the *UP* status, the interval goes back to once every ten seconds. The status of the neighbor gateways is very important to the routing algorithm because the gateway uses this information to determine if there is a possible partition, as explained in more detail in the next section.

For common gateway neighbors, the gateway scans an internal table. When an interface comes up, the controlling process puts the IP address into a table called *ips_tbl*. The common gateway neighbors, however, are not pinged as are the cross net neighbors. The usability of a common gateway neighbor is determined only through the routing update process.

5.1.3 Routing

The route computations are at the heart of the robustness of the RP gateway. In the computations, each gateway-half is considered separately. Currently, all routing updates are done on a periodic basis; each update contains the entire routing table, and there is no acknowledgments for the reception of these data. The routing algorithm lends itself to partial updates and event-driven updates.

Each gateway-half maintains a distance and minimum distance/routing table for both common and cross net routing. Cross-net routing implies that a packet is sent back onto the same local network on which the packet was received. Common-gateway routing occurs when the packet is sent out on a different gateway-half than the receiving gateway-half. This path is the one most often taken. Two routing tables are required to suppress the "triangle" problem where the gateways do not recognize the shorter one-hop path when three networks are connected together in a triangle.

The distance table may contain multiple paths to a destination gateway-half, while the minimum-distance/routing table contains the best path based upon minimum gateway hops. For each destination gateway-half, the next hop gateway, the distance to the destination gateway-half, and the marking of the route is noted in both the distance and routing tables. The "marking" refers to the notation that a given route should be used only if it is the only noninfinite distant route to the destination gateway-half.

The distances are measured in terms of number of gateways traversed to reach the destination. The distance between common-gateway neighbors is defined as 1, but it is changed to infinity if a gateway cannot send traffic on that interface. The distance to a cross-net neighbor is defined to be 0 if that neighbor's status is *UP*. In the case of two common-gateway neighbors interfacing the same network, the distance between these two neighbors is 0.

The algorithm for reducing the distance table to the routing table is based on the following criteria in order of importance:

1. Pick an unmarked route over a marked route
2. Pick the route with the smallest number of gateway hops.

Figure 5.2 is an example of an internetwork system with two PRNETs connected to the ARPANET. Figure 5.3 shows the associated distance matrix from the point of view of gateway B (128.65.20.65⇔128.64.1.51). Figure 5.4 shows the routing tables in gateway B based upon the distance matrix.

As mentioned previously, the status of the gateway neighbors is also reflected

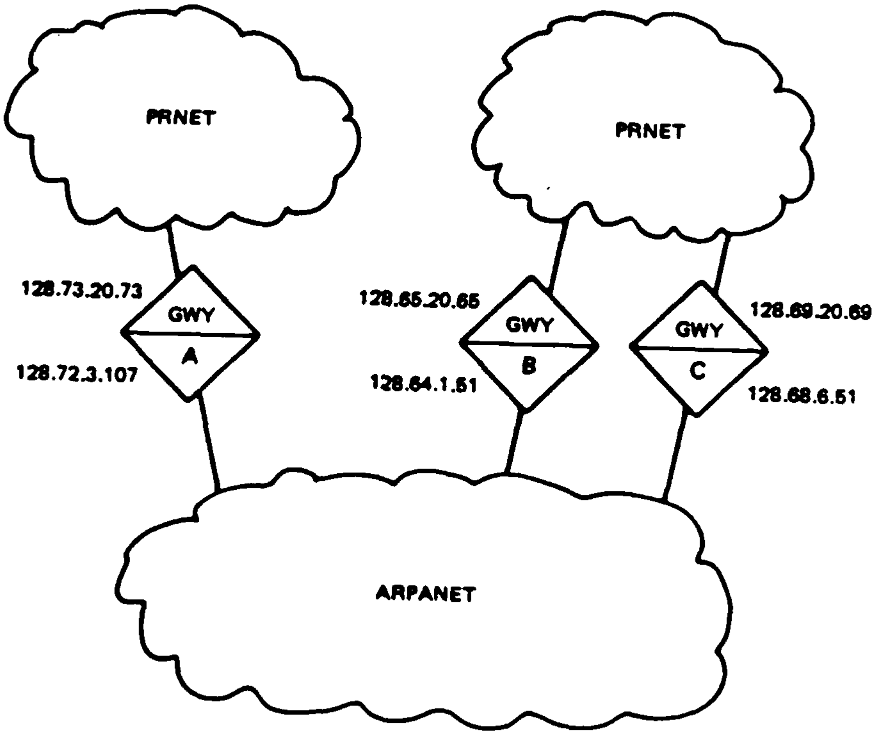


Figure 5.2: Example Internetwork Topology

ARPANET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	0(local)	0	0
128.64.1.51	128.65.20.65	2	0
128.65.20.65	128.65.20.65	1	0
128.68.6.51	128.65.20.65	2	0
128.69.20.69	128.65.20.65	1	0
128.72.3.107	128.65.20.65	2	0
128.73.20.73	128.65.20.65	3	0

ARPANET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	0(local)	0	0
128.64.1.51	128.68.6.51	2	0
128.65.20.65	128.68.6.51	1	0
128.68.6.51	128.68.6.51	0	0
128.69.20.69	128.68.6.51	1	0
128.72.3.107	128.72.3.107	0	0
128.72.3.107	128.68.6.51	2	0
128.73.20.73	128.68.6.51	3	0
128.73.20.73	128.72.3.107	1	0

PRNET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	128.64.1.51	1	0
128.65.20.65	0(local)	0	0
128.65.20.65	128.64.1.51	2	0
128.68.6.51	128.64.1.51	1	0
128.69.20.69	128.64.1.51	2	0
128.72.3.107	128.64.1.51	1	0
128.73.20.73	128.64.1.51	2	0

PRNET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	128.69.20.69	1	0
128.65.20.65	0(local)	0	0
128.65.20.65	128.69.20.69	2	0
128.68.6.51	128.69.20.69	1	0
128.69.20.69	128.69.20.69	0	0
128.72.3.107	128.69.20.69	1	0
128.73.20.73	128.69.20.69	2	0

Figure 5.3: Distance Matrix for Gateway B

ARPANET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	0(local)	0	0
128.65.0.0	128.65.20.65	1	0
128.68.0.0	128.65.20.65	2	0
128.69.0.0	128.65.20.65	1	0
128.72.0.0	128.65.20.65	2	0
128.73.0.0	128.65.20.65	3	0

ARPANET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	0(local)	0	0
128.65.0.0	128.68.6.51	1	0
128.68.0.0	128.68.6.51	0	0
128.69.0.0	128.68.6.51	1	0
128.72.0.0	128.72.3.107	0	0
128.73.0.0	128.72.3.107	1	0

PRNET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	128.64.1.51	1	0
128.65.0.0	0(local)	0	0
128.68.0.0	128.64.1.51	1	0
128.69.0.0	128.64.1.51	2	0
128.72.0.0	128.64.1.51	1	0
128.73.0.0	128.64.1.51	2	0

PRNET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	128.69.20.69	1	0
128.65.0.0	0(local)	0	0
128.68.0.0	128.69.20.69	1	0
128.69.0.0	128.69.20.69	0	0
128.72.0.0	128.69.20.69	1	0
128.73.0.0	128.69.20.69	2	0

Figure 5.4: Routing Matrix for Gateway B

in the distance and minimum distance tables as a mark associated with the route. A route becomes marked when the gateway status of the next hop moves to the *ECHOING* state (i.e., it has missed at least 2 but not more than 4 echoes). If the status of that neighbor goes to *DOWN*, all routes that use that gateway-half as the next hop now have a distance of infinity. Note that if a gateway-half's status is *DOWN* and the destination gateway-half is that gateway-half, its distance never goes to infinity. The route becomes marked but the distance remains 0 in the cross net minimum distance table. This mechanism is used to ensure that the routing algorithm will still try to forward RP control packets, since they are always routed via a cross-net route. Figures 5.5 and 5.6 reflect the changes to the distance and routing tables in gateway B when there is a partition between gateway B and gateway C.

The preference for routing selection is to choose a common gateway path. However, there are three exceptions to this:

1. If a cross-net path is shorter than the common-gateway path and the cross-net route is not marked, the cross-net route is chosen and a redirect is sent if the packet is from a host on the local network.
2. If the cross-net route is not marked and the common-gateway route is marked, the cross-net route will be chosen.
3. If there is only a cross-net path, then the cross-net route is chosen.

If no route is available, an ICMP Destination Unreachable (packet type 3) message is sent back to the host.

When a routing update arrives from a cross-net neighbor any necessary changes are recorded in the common-gateway distance and/or minimum-distance tables. For a common-gateway update, the changes are recorded in the cross-net distance and/or minimum-distance tables. The same reasoning is followed when an update is created. The cross-net routing table is used for common-gateway update and the common-gateway routing table is used for a cross-net update.

5.1.4 Forward Request Protocol

To accommodate mobile hosts, the RP gateway has implemented a forwarding scheme in which the final (or exit) gateway-half re-routes the packet to the host's new internetwork address. The exit gateway-half is the last gateway-half to handle the packet before it is sent to the destination address specified in the IP header.

ARPANET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	0(local)	0	0
128.64.1.51	128.65.20.65	∞	0
128.65.20.65	128.65.20.65	1	0
128.68.6.51	128.65.20.65	∞	0
128.69.20.69	128.65.20.65	1	1
128.72.3.107	128.65.20.65	∞	0
128.73.20.73	128.65.20.65	∞	0

ARPANET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	0(local)	0	0
128.64.1.51	128.68.6.51	∞	0
128.65.20.65	128.68.6.51	1	1
128.68.6.51	128.68.6.51	0	0
128.69.20.69	128.68.6.51	1	0
128.72.3.107	128.72.3.107	0	0
128.72.3.107	128.68.6.51	∞	0
128.73.20.73	128.72.3.107	1	0
128.73.20.73	128.68.6.51	∞	0

PRNET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	128.64.1.51	1	0
128.65.20.65	0(local)	0	0
128.65.20.65	128.64.1.51	2	1
128.68.6.51	128.64.1.51	1	0
128.69.20.69	128.64.1.51	2	0
128.72.3.107	128.64.1.51	1	0
128.73.20.73	128.64.1.51	2	0

PRNET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.1.51	128.69.20.69	∞	0
128.65.20.65	0(local)	0	0
128.65.20.65	128.69.20.69	∞	0
128.68.6.51	128.69.20.69	∞	0
128.69.20.69	128.69.20.69	0	1
128.72.3.107	128.69.20.69	∞	0
128.73.20.73	128.69.20.69	∞	0

Figure 5.5: New Distance Matrix for Gateway B

ARPANET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	0(local)	0	0
128.65.0.0	128.65.20.65	1	0
128.68.0.0	128.65.20.65	∞	0
128.69.0.0	128.65.20.65	1	1
128.72.0.0	128.65.20.65	∞	0
128.73.0.0	128.65.20.65	∞	0

ARPANET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	0(local)	0	0
128.65.0.0	128.68.6.61	1	1
128.68.0.0	128.68.6.51	0	0
128.69.0.0	128.68.6.51	1	0
128.72.0.0	128.72.3.107	0	0
128.73.0.0	128.72.3.107	1	0

PRNET Common Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	128.64.1.51	1	0
128.65.0.0	0(local)	0	0
128.68.0.0	128.64.1.51	1	0
128.69.0.0	128.64.1.51	2	0
128.72.0.0	128.64.1.51	1	0
128.73.0.0	128.64.1.51	2	0

PRNET Cross-net Gateway Matrix

Dest. Gwy-Half	Next Hop	Distance	Mark
128.64.0.0	128.69.20.69	∞	0
128.65.0.0	0(local)	0	0
128.68.0.0	128.69.20.69	∞	0
128.69.0.0	128.69.20.69	0	1
128.72.0.0	128.69.20.69	∞	0
128.73.0.0	128.69.20.69	∞	0

Figure 5.6: New Routing Matrix for Gateway B

The RP gateway receives forwarding information in the form of a host forward request packet sent from the host after affiliation with a new gateway. When a gateway-half receives this packet, it clears all information regarding this host ID from its forwarding tables; if necessary, it also informs all cross-net neighbors to remove any information regarding this host as well. This is required to prevent routing loops in the case where the host reaffiliates with one of its previously affiliated gateways.

If the host forward request packet is empty, (the field which contains the number of addresses equals 0), the gateway immediately returns a FORWARD-REPLY packet. However, if the host used to be affiliated with a different gateway-half, the text of the packet contains all of the active old addresses of the host. When a gateway-half receives this information, it generates a gateway-to-gateway forward request packet, and sets the destination of the IP header to the old host address. The exit gateway-half is responsible for intercepting this packet, inserting the information into its forwarding tables, and broadcasting this information to all of its cross-net neighbors.

The host's FORWARD-REQUEST packet is retransmitted by the host until a host FORWARD-REPLY is received from the gateway that received the request. This is to ensure that this information has been delivered to all required gateways. The exact time at which the reply is transmitted to the host varies depending on whether the old address is on the same physical network as the new address. If this is true, a host FORWARD-REPLY is sent immediately to the host. If the new host address is located on a different network, the gateway-half intercepting the packet sends a gateway-to-gateway forward reply, ICMP packet type 24, to the gateway that sent out the gateway-forward-request packet. The gateway-half who receives this reply then sends a host-forward-reply packet to the host who originated the host-forward-request packet. After a gateway processes a forward request, either generated by a host or forwarded by a gateway, it broadcasts this information to all of its cross net neighbors as a gateway to gateway forward request packet. This forward request, however, is sent as a RP packet, IP packet type 9, and is retransmitted three times in 30-second intervals to improve its probability of successful delivery.

There is a separate acknowledgments table for each gateway-half. Whenever a gateway receives a host forward request packet, it places the request in this table and sets an acknowledgments field appropriately. If a host forward reply has been sent out, this field is marked true; otherwise, it is marked false. To ensure the gateway retains the latest information regarding each forward request, sequence numbers are used in all forward request and reply type packets. A host forward request is always processed by a gateway since it is assumed that the host always has the most up-to-date information. An ICMP gateway-to-gateway forward request or reply packet is processed only if the sequence number is greater than the last packet processed.

Whenever a gateway-half receives a packet, it checks to see whether the packet needs to be forwarded. If it does, the gateway copies the destination address from the IP header into the option field of the packet (using the OLD DESTINATION ADDRESS option) and places the correct host address in the IP header destination address field. After forwarding the packet, the gateway sends an ICMP READDRESS to the source informing it that it should send all packets destined for the old address to the new address. In general, the source host can then quickly take over the readdressing of these packets, and the gateways need not be burdened by forwarding.

5.1.5 Exterior Gateway Protocol

The Exterior Gateway Protocol (EGP) is a mechanism for the exchange of network reachability information between autonomous gateway systems. One of the motivations for the design of EGP was to allow exchange of routing information between the gateways running differing routing protocols and route computation schemes, such as the RP gateways and the other ARPANET gateways.

To provide for eventual EGP compatibility between the RP and IP gateways, the address structure of RP addresses was designed to be backwards compatible and meaningful to IP gateways. Thus the unique host IDs are restricted to 14 bits (16,383 maximum hosts), which is clearly too small in general.

We implemented EGP software for the gateway and an EGP test program that runs on TOPS-20 systems to test our implementation. However, because of address-space and memory limitations of the LSI-11 processor, we were not able to integrate the EGP software into the RP gateway. This limitation did not affect the proposed experiments.

5.2 Terminal Interface Unit Modifications

The terminal interface unit (TIU) is a unit designed to act as a terminal controller to attach terminals to the PRNET and a variety of other networks. The TIU was one of the first implementations of TCP/IP and thus has a strong or'entation to communications support. The TIU was designed in a layered and modularized manner; as a result, most of the RP changes were confined to a single module. The modifications to support RP were fairly straightforward and are described below.

5.2.1 TELNET

At the highest level, no changes were required in the TELNET terminal handler software. Since the use of name servers is an important part of the RP design to handle the long-term dynamics of maintaining the host name-to-address translation table, the TIU TELNET will eventually need to access name servers rather than using internal tables. For the purposes of the experiments, the TIU operators used numeric addressing.

5.2.2 TCP

Only minor changes were required in TCP, which reflected earlier implementation design decisions taken when TCP was first coded. TCP normally calculates its maximum data segment size based on the maximum IP segment size and knowledge of the amount of IP header options, if any. Since the addition of the TCP source and destination identifiers is done at the IP level, TCP was changed to ask IP the size of the maximum data segment.

Since TCP (and IP) already performed fault-isolation to force re-routing upon local gateway failure, no other changes to TCP were necessary.

5.2.3 IP

The changes to IP were also minor. The processing of the TCP identifier options was placed in an expanded ICMP module to contain all of the RP-specific code to a single file. The IP software was only modified to call this options handling routine. IP was also modified to only check the unique host identifier of the RP address (low 16 bits) when validating that received packets were intended for this TIU.

Because IP called an ICMP routine to select the local-network destination for each packet, the significant changes and problems encountered in modifying Unix, discussed later, were absent.

5.2.4 ICMP

The vast majority of the RP-specific changes were in the ICMP module. This module handles the transmission and reception of all ICMP messages and manages the redirect and local-network destination tables. This organization made the changes to support RP much easier than would an organization, such as in

Unix, that placed ICMP and fault-isolation functions throughout the networking code.

The actual list of changes to ICMP are too extensive to discuss in detail. However, in general the changes mirror the protocol requirements:

- Implementing the host-gateway protocol, including getting GID information.
- Processing of old-address IP options.
- Storing new-address information in the destination-specific information table.
- Improving the code to handle host-specific redirects.
- Adding a table to store the TIU's currently active previous addresses.
- Significant improvement to the status display routines to notify the TIU user of significant RP status information and to display internal ICMP tables.

5.3 VAX/Sun Unix

In addition to the TIU, a full-function host was needed to demonstrate the use of RP protocols; for a variety of reasons such as size, weight, and power for the airborne host, Unix operating on VAX (ground-based) and SUN (airborne) hardware was selected.

Since the total amount of new functionality of RP is small compared with the total functionality of IP, RP was implemented as modifications to the existing Unix 4.2bsd network code. Experience with the standard TCP/IP code in 4.2bsd revealed a number of areas in which the protocol was not implemented completely: (1) no fault isolation of local network destinations; (2) TCP connections are automatically closed after a constant level of non-connectivity; (3) receipt of ICMP destination unreachable messages forced closure of any TCP connections routing via that destination; (4) ICMP redirect messages will force the re-routing of only one connection at a time.

Unfortunately, the areas in which the 4.2bsd TCP/IP code fell short lay precisely in areas critical to the correct operation of RP–robustness and survivability. Because of these problems, we evaluated a version of TCP/IP implemented by BBN as a functional replacement for the 4.2bsd code. Tests showed that this new code solved most of the problems we had observed during early non-RP

SAC experiments in the Berkeley version. Thus we selected the BBN TCP/IP kernel as the basis of the Unix work.

The implementation of RP in the BBN TCP/IP for 4.2bsd Unix consisted entirely of modifications made at the IP and interface driver levels. No changes were made to the code for TCP, and only minor changes were made to any other higher-level protocols.

5.3.1 Interface Drivers

In 4.2bsd, the network address from each interface is maintained by each of the network interface drivers, leaving IP/ICMP somewhat network-independent. Thus the gateway affiliation mechanism is implemented in each of the network drivers; this approach works well in that the PRNET mechanism is slightly different from the ARPANET mechanism, because the ARPANET did not support logical addressing.

Two new Unix IOCTL calls are provided; SIOCSTARTRP IOCTL is called to start the affiliation mechanism running on an interface, while SIOCSTOPRP IOCTL stops it.

The code that implements the "set interface address" IOCTL (SIOCSIFADDR) was modified to call the affiliation module in addition to setting an interface's address. This change allows the affiliation to handle a user-initiated address change in the same way as an address change resulting from internetwork dynamics.

To support multicasting of affiliation requests on the ARPANET, a multidestination addressing facility was implemented in the driver to sequentially send the affiliation requests to each address in a gateway list. This list is maintained by a new IOCTL.

Finally, the ARPANET driver was enhanced to support class-B addresses (the form that RP addresses take) in addition to the normal class-A addresses.

The interface code maintains a data structure (called *ifnet*) that contains:

- Active interface IP address (may be null)
- List of previous IP addresses (may be null)
- Unique host (interface) ID (HID, may be invalid but not null)
- Local net address (LNA, can be mapped from HID, can be null)
- Physical network number (may be null)

- Affiliation state
- Flags.

5.3.2 Routing

The most complex and troublesome changes to the Unix TCP/IP centered around routing. Unlike the TIU, which performs a route look-up or computation for each packet within the IP layer, Unix optimizes time spent in routing by performing the route computation once and then binding this route to the high-level connection. Thus routing information is distributed throughout a number of data structures rather than centralized within a single table in IP. This distribution of the information made it particularly cumbersome and time-consuming to implement the RP changes. The changes made in routing were similar to those made in the TIU.

The biggest problems in modifying Unix were the result of the network software trying to play a very active role in routing of packets rather than handing this task over to the gateways. Unix has implemented a variety of schemes that had to be defeated to ensure that the gateways control the traffic routing. For example, Unix will not always accept a redirect from a gateway, preferring to believe its internal routing table. The mechanisms that implemented load-sharing across multiple local-network gateways also had to be defeated.

Using the standard Unix nomenclature, an "interface route" to the interface's attached physical network is added when the set interface address ioctl is called. When an interface is taken down, all routes via that interface are deleted, and all connections bound to the deleted routes are rebound.

When a new affiliation is made, a "gateway route" to "default" via the affiliated gateway is added to the routing table. The "gateway route" to "default" via the previous affiliated gateway is deleted, and connections bound to this route are rebound. An "interface route" to the affiliated gateway's GID is added. Note that the result of an "interface route" existing to the affiliated gateway's GID and to the interface's physical network is that packets to destinations affiliated with the same gateway and destinations on the same physical network are sent directly, not to a gateway.

5.3.3 IP Packet Handling

The actual handling of IP packets is changed very little for RP. On output, if either the source or destination address(es) are found in the changed address list, the original source and/or destination address (as appropriate) is moved into an IP option and the new address(es) are inserted into packet header. Then, if the bound route does not match the new destination address in the IP header, the connection is rebound to a route.

On input, if either of the two special RP IP options is present, the options are processed by moving the address from option field into the IP header. The IP module was also modified to accept all packets with an IP destination address equal to any of the active or previous IP address of any interface; all other packets are passed to *ip_forward()* as usual.

5.3.4 User-Level Code

The only changes to upper-level protocol software involved the processing of network status information returned from TCP/IP. In general, the TCP and IP layers do not filter status information but provide it "raw" to the higher-levels to deal with. This is unfortunate in that the higher-layers often do not know the context of the status message. For example, a TCP retransmission time-out (excessive number of retransmissions) that occurs while searching for a new local net gateway does not necessarily indicate anything about the remote host status; only that communications was disrupted. In this case, the time-out should be ignored until the communication path is reestablished or a failure to find a gateway is declared.

For the purposes of the experiments, the TELNET user and server (daemon) programs were modified to set the TCP retransmission timeout to infinity as well as to ignore ICMP Unreachable messages. While these changes to TELNET are not optimal, they were effective. In general, the network code must provide better information to higher-level protocols as to what the problem is and whether it is a short-term transient, longer-term problem, or "permanent" failure.

Chapter 6

Reconstitution Experiments

The final step of the RP work consisted of tests and experiments with the system automatically adapting to a variety of network partition and reconstitution problems. While the experiments were a final logical step, in reality, testing and experimentation was conducted in parallel with the implementation to allow us to experiment, test, and verify basic RP functions and then to build upon these functions in later experiments.

The resulting experiment list, ordered by increasing protocol functionality or complexity, is:

- F5/10–Partitioning and Coalescing of PRNETs
- F8/11-ARPANET and multiple network partitions
- F9–Network Mobile Host.

6.1 PRNET Partitioning and Merging–F5/10

The F5/10 demonstration was conducted on September 9, 1985, and was the first of three demonstrations planned for the reconstitution protocol. The new protocols shown in F5/10 are required by an operational user tc enable single networks to be automatically formed from multiple ones and to enable the user to automatically reroute information when the network on which he is working becomes partitioned (fragmented). The F5 demonstration accomplished automatic rerouting when a partition occurred in a (single) PRNET. The F10 demonstration showed automatic rerouting of computer information when two PRNETs were coalesced.

The initial conditions for conducting the F5/10 demonstration, shown in Figures 6.1 and 6.2 were to have both the PRNET and ARPANET links available. When this condition existed, a link was opened from the user terminal/TIU (SRI laboratory at Offutt AFB) through the aircraft relay to the VAX computer host at Camp Dodge, Iowa. This link was established and a data file was read to the user terminal. The rate of flow of information was pointed out to observers and the lack of activity on the gateway monitor (due to the direct PRNET link) was emphasized. To confirm RP operation the aircraft PR was turned off and automatic rerouting of data due to gateway redirection occurred within about 1.5 to 2 minutes via the PRNET. During the time traffic was being routed via the ARPANET, the gateway monitor activity was very high as it provided a line of data for each packet that was being transferred through it.

This procedure was successfully done three or four times before we extended the demonstration to what would happen when the aircraft approached the edge of the line-of-sight transmission range. In this extension of the demonstration the effects of short-term (seconds) dropouts, multipath on the aircraft, and shadowing were to be tried. The RP responded to this stressed environment and operated just as when the aircraft PR was turned off/on.

This demonstration showed several components of the protocol in operation, the most important being the automatic affiliation of a host (TIU or VAX) with a gateway and routing between gateways based only upon gateway identifier, not network number.

In addition to all the software functionality of a standard internetwork gateway, the following internal mechanisms specific to the reconstitution protocol were exercised:

- Hosts using PRNET logical addressing to find gateways attached to a PRNET, eliminating the need for built-in tables or preassigned addresses.
- Hosts taking part of their address dynamically from their affiliated gateway rather than using a statically defined network number.
- Gateways dynamically determining the existence of other gateways on a PRNET rather than using built-in tables.
- Gateways exchanging gateway-centered routing table information.

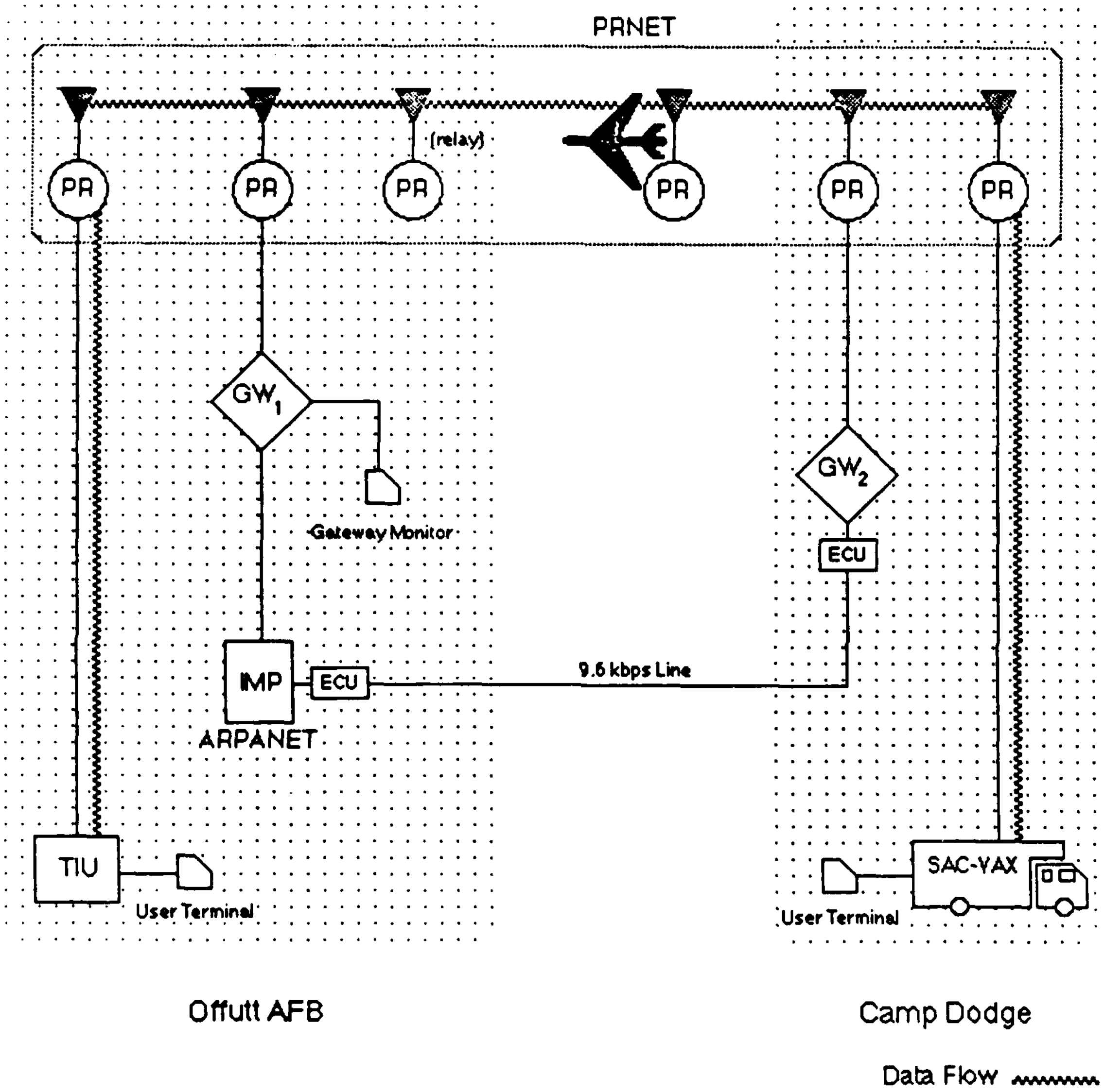


Figure 6.1: F5/10 Demonstration—Coalesced PRNETs

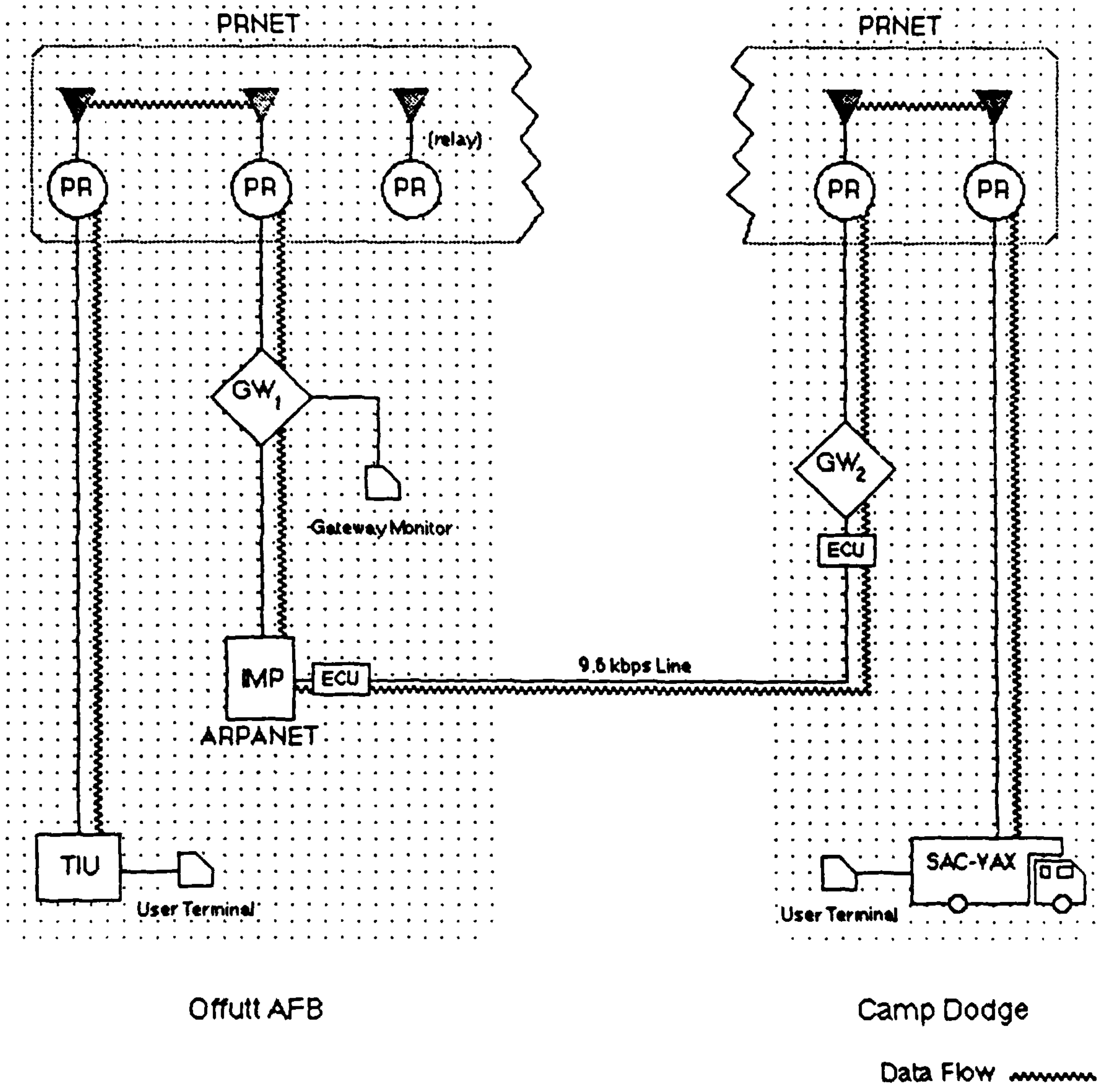


Figure 6.2: F5/10 Demonstration—Partitioned PRNETs

Further details on the experiment, including equipment setup and a detailed scenario, may be found in the experiment report [11].

6.2 ARPANET and Multiple Network Partitioning–F8/11

The F8/11 demonstration was conducted on February 27, 1986, and was the second of the three demonstrations planned for the reconstitution protocol. This demonstration showed several new components of the protocol in operation. The most important of these is the host (VAX) automatic affiliation with an ARPANET gateway and rerouting between gateways through multiple partitions. The actual demonstration was conducted in the SRI laboratory network at Offutt AFB and consisted of two different configurations: F8, with the VAX on the ARPANET, and F11 with the VAX on the PRNET.

Although the F8 and F11 demonstrations differed in the attachment of the VAX, the rest of the demonstration test set-up was identical. The initial conditions for conducting the F8 demonstration are shown in Figure 6.3; the extra PRNET nodes and gateways required for F11 are not shown on this figure since they did not participate in F8.

A connection was established from the SAC-VAX at Offutt AFB to the SRI-JOYCE VAX at SRI and, as is usual in these demonstrations, a data file was read to the user terminal. The rate of flow of information was pointed out to observers and the lack of activity on the gateway monitor (due to the direct ARPANET link) was emphasized. To confirm RP operation the truck line that connect ARPANET node IMP-3 with IMP-80 was removed, effectively partitioning the network into the remainder of the ARPANET and a single-node ARPANET. Automatic rerouting of data due to gateway redirection occurred within about 1.5 to 2 minutes via the PRNET; Figure 6.4 shows the data flow. During the time traffic was being routed via the PRNET, the gateway monitor activity was very high as it provided a line of data for each packet that was being transferred through it. The final step of the demonstration involved reconnecting the trunk line, effectively restoring IMP-3 to the ARPANET; the traffic was redirected back over the direct ARPANET path within about 2 minutes. This procedure was successfully done several times before reconfiguring for F11.

To conduct F11, the SAC-VAX was moved[1] from the ARPANET to the PRNET, and the ARPANET IMP-3 was disconnected from the rest of the

[1] Because the ARPANET and PRNET have different network interfaces and drivers, the machine was rebooted with a new kernel. A multi-network driver in the VAX would have allowed transparent reconfiguration.

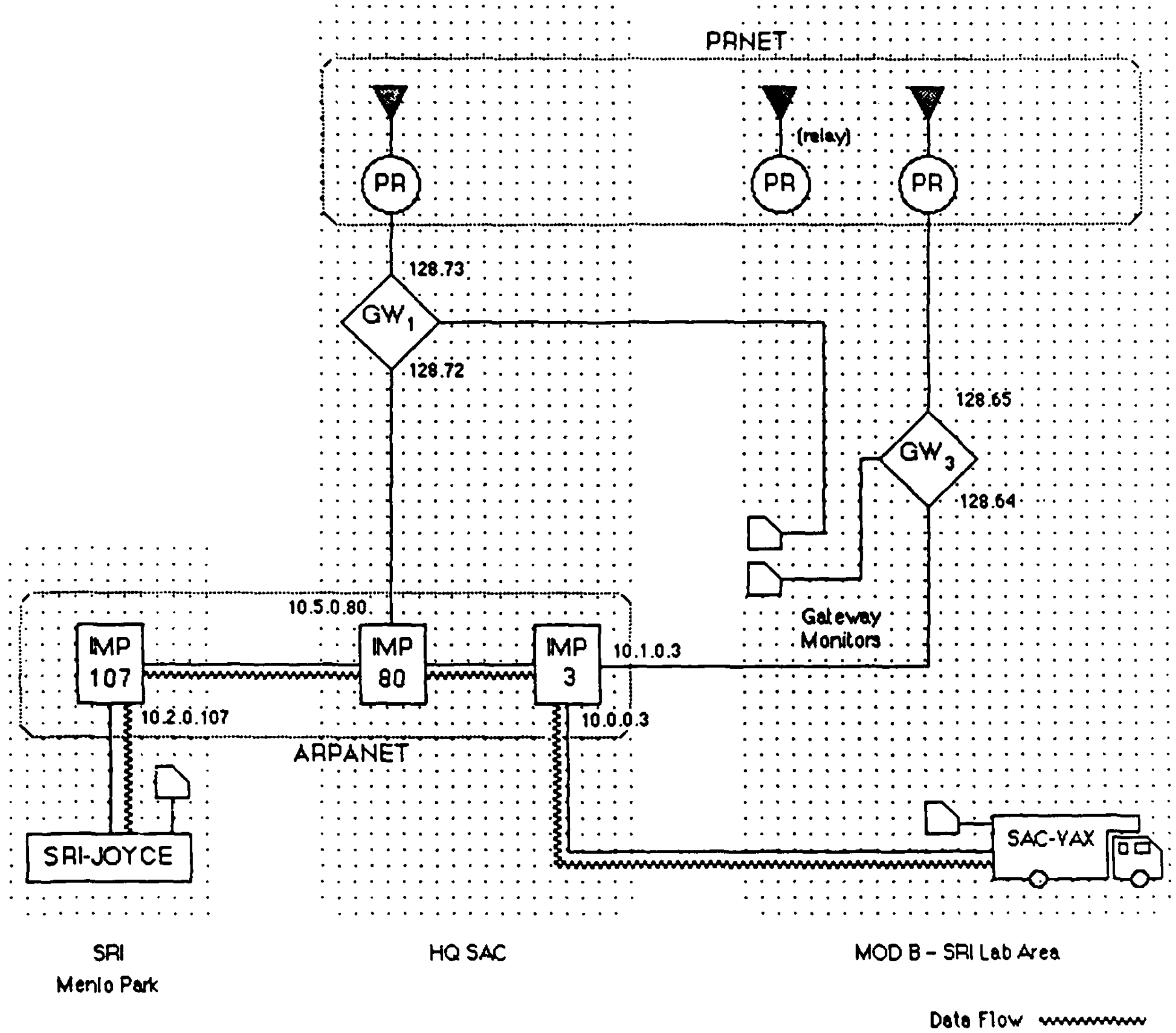


Figure 6.3: F8 Demonstration–Initial Configuration

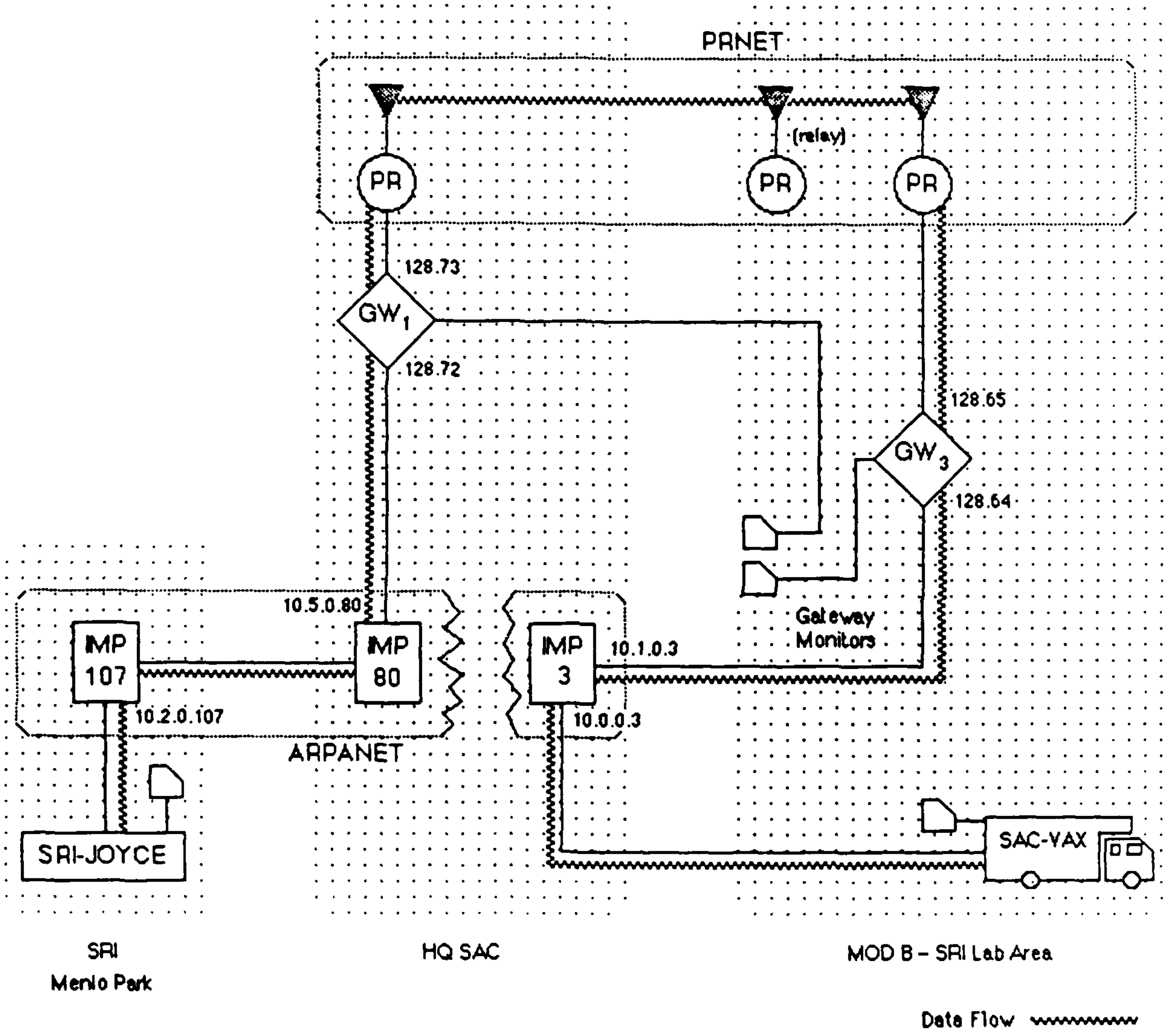


Figure 6.4: F8 Demonstration–Partitioned ARPANET

ARPANET as described for F8. After establishing a connection to SRI-JOYCE, with the resulting data flow as shown in Figure 6.5, the PRNET was partitioned by turning-off a repeater that provided connectivity between the two PRNET segments connected via coax cable. Within 2 minutes, the traffic resumed, and packet logger activity on the gateway consoles showed that the traffic was traversing the three gateways as shown in Figure 6.6.

This demonstration showed several components of the protocol in operation, the most important being the operation of the VAX on the ARPANET with RP software, and routing through several gateway hops. In addition, operation of F8/11 showed users that they should really not be overly concerned with the "exact" route their traffic takes; with automated rerouting they should let the networks handle the dataflow as much as possible. This point was aptly demonstrated in the F11 demonstration, when, as we reconfigured four networks back into two, and after all nodes were interconnected, data was noted passing via one route while acknowledgments were travelling a different one. It was also observed that when partitions were eliminated in the ARPANET (F8), there was no break in traffic flow as the PRNET handed traffic flow back to the "coalesced" ARPANET.

Further details on the experiment, including equipment setup and detailed scenario, may be found in the experiment report [12].

6.3 PRNET Network Mobile Host–F9

The third and final RP demonstration was conducted on June 27, 1986. This demonstration of the mobile host capability culminated the RP development efforts by showing a command element equipped with a processor (host) moving among various network ground-entry points (gateways) while that host automatically establishes and maintains connectivity with the other command center hosts.

The demonstration was conducted from the SRI laboratory area on Offutt AFB, Nebraska. Additional sites participating were Camp Dodge, Iowa, where a gateway between the PRNET/ARPANET was located, a remote (fixed) host (VAX 11/750) at Menlo Park, California, and an airborne command post (ABNCP) aircraft equipped with a mobile host (SUN 2/170) and Packet Radio (PR).

Two PRNETs were established, one at Offutt AFB and the other at Camp Dodge, and linked via RP gateways to the ARPANET. Both PRNETs were operated on the same frequency parts so the airborne PR could establish con-

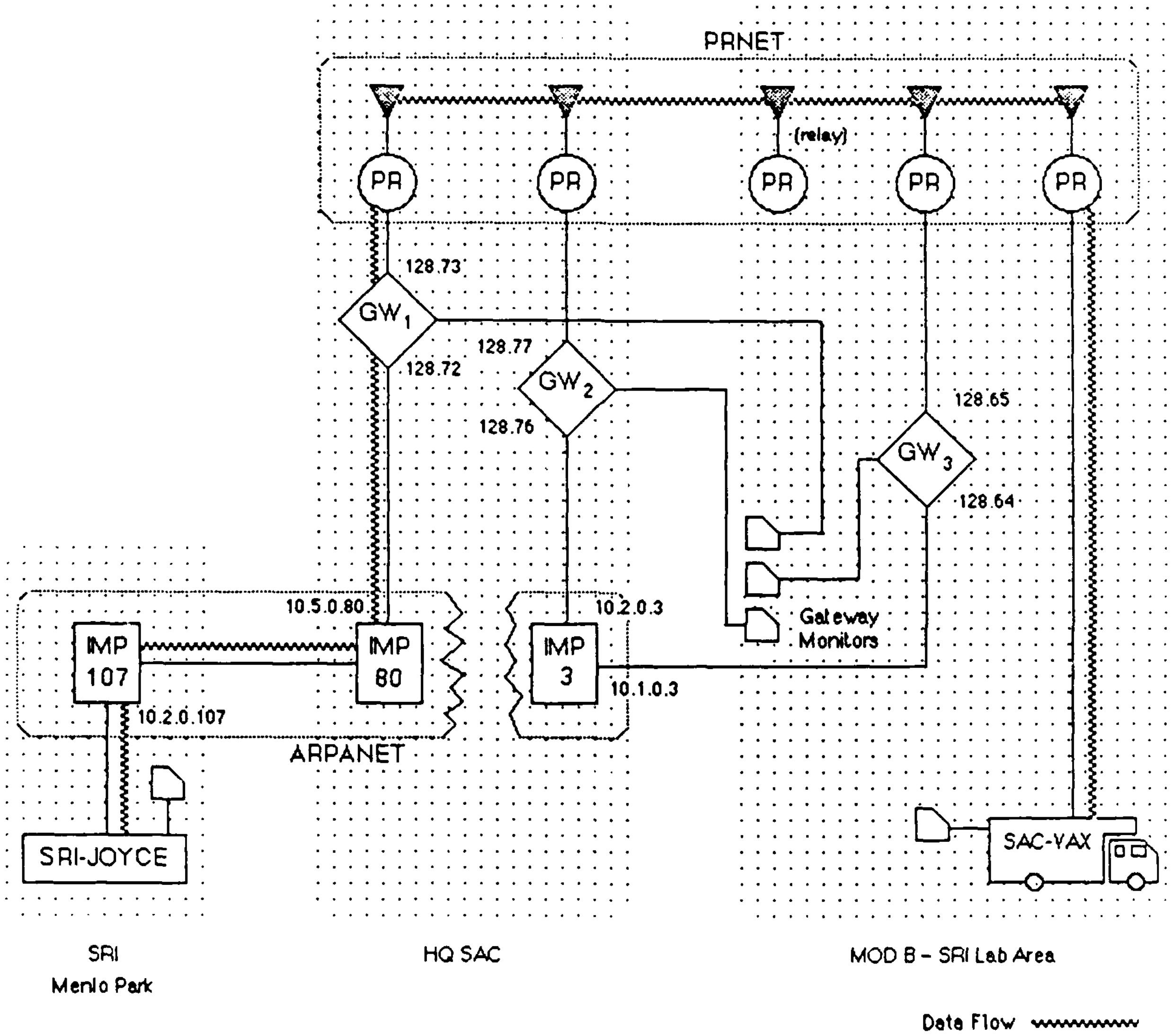


Figure 6.5: F11 Demonstration–Partitioned ARPANET

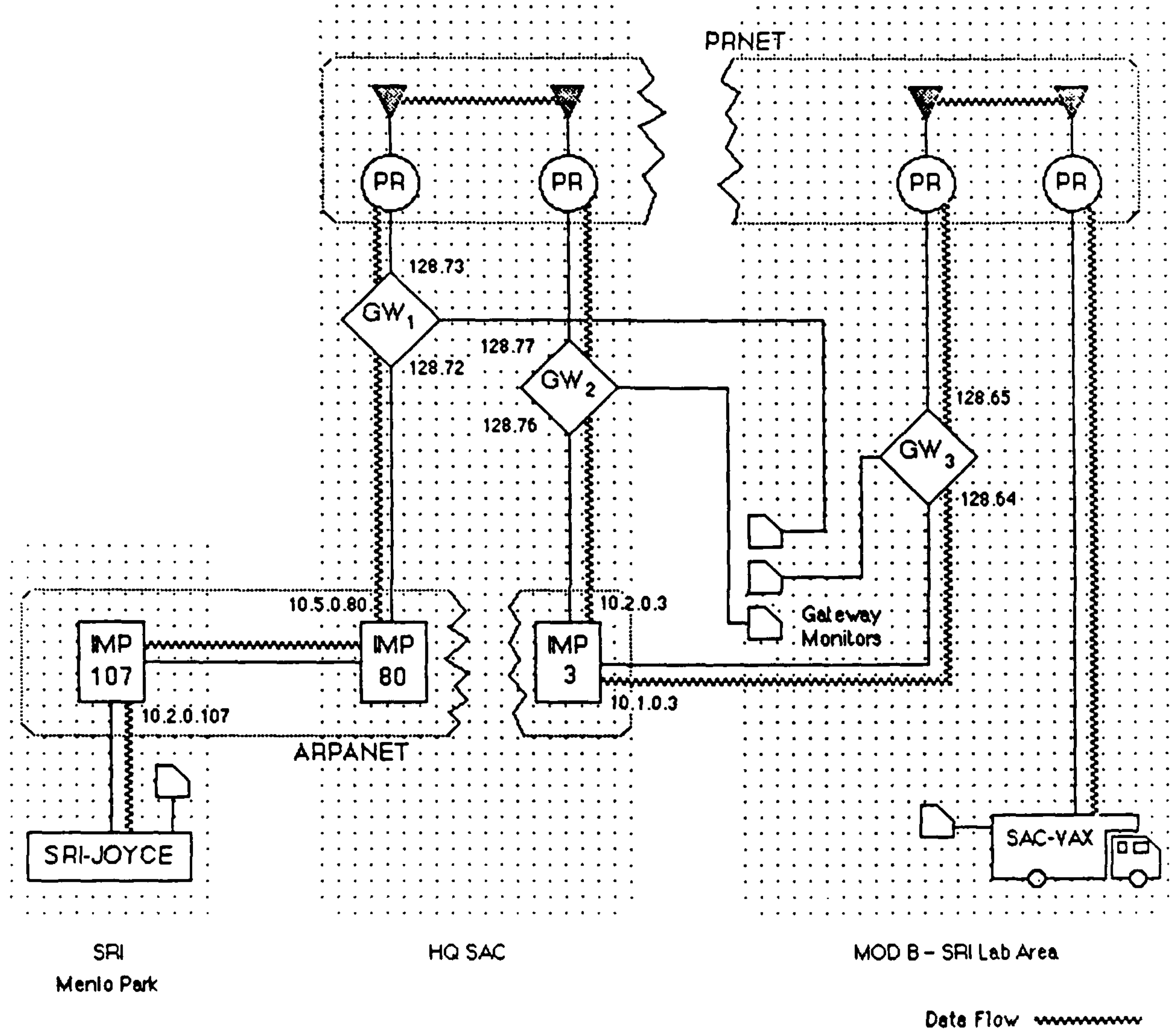


Figure 6.6: F11 Demonstration–Partitioned ARPANET and PRNET

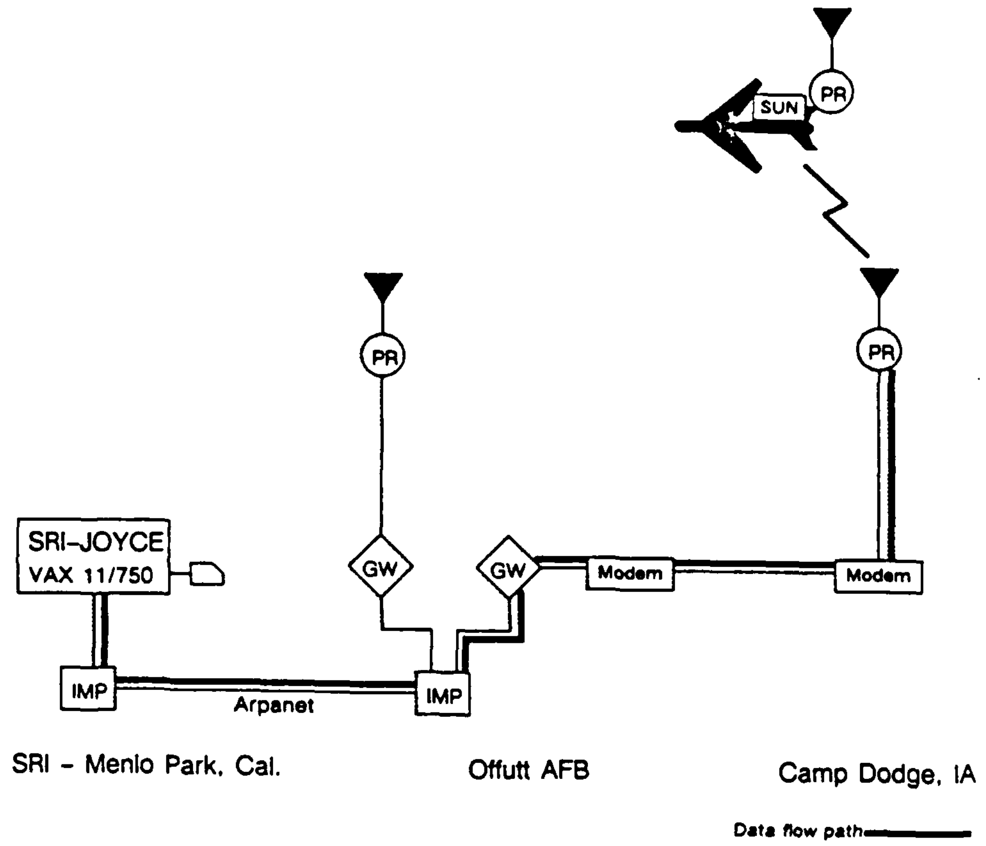


Figure 6.7: F9 Demonstration—Network Mobile Host

nectivity through whichever was within line-of-sight.

The demonstration was conducted in two stages. While the plans were to have the airborne host make initial contact with the PRNET based at Offutt, weather problems precluded starting the experiment until the aircraft was in range of the Camp Dodge PRNET. A data transfer of a file residing on the airborne host was initiated from the SRI-JOYCE computer; Figure 6.7 shows the data flow paths.

Once the transfer was started, the aircraft flew towards Offutt. Again, the plan was to have the airborne PR merged the two networks into a single network (as in the F5/10 demonstration); however, the flight path and antenna patterns were such that there wasn't an overlap in coverage. Eventually the airborne host flew out of range of the Camp Dodge PRNET and the traffic flow ceased. Approximately five minutes later, the aircraft flew within range of the Offutt AFB PRNET and the airborne host automatically switched to the ground-entry point (gateway) at Offutt.

This final demonstration exercised most of the RP software shown in previous demonstrations and new capabilities that allowed resumption of data flow following changes in addresses. The specific features shown included:

- Gateways exchanging routing information across multiple networks (also shown in F5/10 and F8/11).
- Hosts dynamically affiliating with gateways (also shown in F5/10 and F8/11).
- Hosts reaffiliating with a new gateway after partition/failure/movement.
- Hosts sending and gateways processing FORWARD-REQUEST packets.
- Gateways forwarding of data packets following relocation of a host.
- Hosts processing ICMP READDRESS packets.
- Hosts handling old-source and -destination address options.

Further details on the experiment, including equipment setup and a detailed scenario, may be found in the experiment report [13].

Chapter 7

Conclusions and Observations

The intent of the reconstitution experiment effort was not to construct a final, operational system, but to explore issues and new architectures and to gain an understanding of the problems associated with internetwork dynamics. To that end, we discuss here unresolved problems, unanticipated problems, and areas in need of further work.

7.1 Functionality

In terms of functionality (without regard for performance), the RP protocols performed as expected in that handling of partitions and merges was an automatic consequence of changing to a gateway-centric organisation. The packet-forwarding functionality to handle network mobile hosts also worked as expected; however, this area is a complex one involving trade-offs and scalability issues, which were not adequately explored in our small test network.

An unexpected problem of the dynamic, gateway-centric organisation occurs when a single gateway into a network fails and is replaced by a hot-standby gateway (with a different gateway ID). In this case, all of the hosts on that "stub" network must change their address, but there is no operational "previous" gateway or previous gateway cluster to handle the packet-forwarding functions. We have derived a solution to this problem as described in [10]; but what happens when you add the complication of a host moving from the stub-network to another network at the same time as the gateway failure? The proposed solution was not adequately exercised in testing and, in any event, probably leads to a variety of anomalous cases such as described above. The best solution may be to note the problem and suggest that the replacement gateway have the same gateway ID as the failed gateway.

A final area of functionality that we could not test involved dynamic name servers. A host must register its new address with the name server whenever it changes gateways because of a finite holding-time of information in the gateway forwarding tables. However, existing name servers do not allow an automated means to update the name-address translation tables, primarily because of authentication issues. The dynamic addresses remove the last possibility of relying on fixed addresses to perform authentication functions; currently, some amount of verification can be performed by checking the human-maintained tables versus address. Additional authentication mechanisms must be developed and implemented before switching to a dynamic, gateway-centered approach is feasible from a total system perspective.

7.2 Performance

While the RP protocols functioned correctly and recovered from the various faults tested, the delays exhibited in the systems, which are equivalent to delays in the existing internetwork, are probably excessive for operational use in a stressed environment. Additional work is required in link protocols and routing algorithms, independent of the issues of network reconstitution, to improve the responsiveness of the internetwork to rapid or massive changes in topology.

The RP system was designed as an extension to the current internetwork that was as backward compatible as possible. This approach was motivated by the pragmatics of implementing changes to gateways and hosts, particularly the changes to Unix, and the desire to interoperate with unmodified hosts on a limited basis. Accordingly, the RP system exhibits many of the same implementation deficiencies in regard to responsiveness that are found in the current system. This lack of performance typically arises from a loose coupling of the internetwork gateways functions from the underlying network control functions. When failures and dynamics are assumed to be the exceptional condition, delays of 1 to 2 minutes are acceptable when compared with the delays incurred with manual intervention. However, when dynamics is the rule, the performance of the internetwork must be improved significantly over that exhibited today in both the standard and RP systems.

The time that the RP gateway takes to respond to a change in network topology, in general, depends on a variety of timers, time-out values, and link parameters. For example, in the case of a network partition affecting intranet traffic, the sending host must first fault-isolate the destination host as having a problem (30 to 60 s); at the same time, the local gateway must lose contact with the gateway in the other partition (60 to 90 s) in order to trigger a route recomputation. Thus we observed response time on the order of 1 to 2 minutes. These delays arise from using simple link up/down models to determine connectivity; with a 15-second polling interval and four missing packets to bring a link down, minimum response time is 1 minute. These times are somewhat optimistic, because they result from time-out values that are probably too short for general use; several times we have observed ARPANET partitions detected when, in fact, the problem was temporary congestion on the cross-country lines.

For a generally static environment, these times are probably acceptable and in any event are comparable to the response times of the existing internetwork system. But for the environment where reconstitution is likely required, these times are too long. Additional work is needed in the design of network structures with fast response time and acceptable overhead.

7.3 Overall Architecture

Overall, the resulting RP architecture satisfies almost all of the design constraints and is a logical improvement over the existing architecture. The concept of "exit gateway" has been adopted by BBN for use in the next version of the internetwork gateway; routing based on exit gateway is the first step toward implementing RP routing.

As noted earlier, the concepts behind dynamic binding of internetwork address and transport connection identifier are becoming generally accepted.

7.4 Ease of Implementation

One of our desires was to impact the hosts as little as possible and to provide a degraded mode of operation so that unmodified hosts (such as the IPLIs) could still function. As indicated earlier, the modifications to the TIU were accomplished very simply, involving only minor changes to the IP handler and implementation of the host-gateway protocol in the ICMP module. Perhaps the simplicity of the TIU changes was no accident, in that we unconsciously designed the RP protocols to easily integrate into the TIU.

However, the modifications to the Unix protocols were surprisingly complex, costly, and error-prone; in fact the Unix testing was almost as time-consuming as tests to the gateway themselves. The difference between the TIU and Unix was only in the way the routing procedure was handled. However, the Unix had a very complex, "optimised," and network "smart" routing system that in many cases acted to defeat the actions of the gateways. Perhaps this shows what happens when a host attempts to perform routing functions that rightly belong in the gateway.

Appendix A

Host-Gateway Protocol

One of the significant changes to the internetwork architecture is an explicit protocol that is executed between the host and its gateway. This protocol is the cornerstone for many of the reconstitution protocol features. To explain the host-gateway protocol, we will first describe the general interaction model, for both complete and minimal implementations, and then the network-specific aspects for the PRNET and ARPANET.

A.1 General Interaction Model

To support the network reconfiguration dynamics while maintaining efficiency, the RP system is organized around gateways rather than around networks. Rather than having a fixed, assigned network number as part of its address, the host's address contains a field that identifies the gateway "closest" to the host. For interoperability with the existing DoD internetwork hosts and gateways, the gateway ID is allocated 14 bits and the host ID is also allocated 14 bits, so that gateways can be addressed as hosts in a similar manner.[1] The resulting 32-bit IP address is thus:

[1] While this bit allocation is inadequate for a large internetwork system with many networks, gateways, and hosts, it is sufficient to conduct our experiments in network reconstitution, and increasing the size of the fields has no impact on the operation of the protocol.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|   Unique Gateway ID       |0 0|   Unique Host ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

To the current internetwork system, this 32-bit address appears to be a Class-B address with the host attached to the Class-B network identified by the gateway ID. This address definition is compact and is compatible with existing internetwork addresses, and allows interoperations of the reconstitution gateways with the current internetwork system via the Exterior Gateway Protocol (EGP).

Each host is thus considered to be associated with a gateway in much the same way that an ARPANET host is associated with an IMP. Unlike the ARPANET, the binding between host and gateway is dynamic, and the host-gateway protocol is the mechanism that maintains this binding.

The host-gateway protocol logically comprises four stages:

Step 1–Gateway Search The host conducts a search for potential gateways on the attached network with which to associate. Since the network topology can radically change, built-in tables of gateway addresses are not sufficient. The search is conducted by network-specific mechanisms defined for each network, usually using logical addressing or broadcast mechanisms.

Step 2–Address Binding Once a gateway is found, the host constructs its current address from its unique ID and the gateway's unique ID. If a gateway is not found after a diligent search, the host uses the value "0" in place of a gateway unique ID, while still trying to find a gateway.

Step 3–Forwarding Notification The host next notifies the new gateway of all of its previous addresses, if any, that are currently active. The gateway uses this information to establish forwarding tables in the previously associated gateways.

Step 4–Status Monitoring Finally, the host monitors the state of its associated gateway. If a problem is detected, the host drops its association with the gateway and reverts to the gateway-search mode.

In certain situations, it would not be possible to implement a complex host-gateway protocol. In particular, some systems (such as the IPLI) that need to work with the reconstitution gateway cannot be changed. These considerations led to the use of the ICMP ECHO and ECHO-REPLY packets as the basis of the gateway acquisition procedure and to the separate specification of additional forwarding addresses in the FORWARD-REQUEST message.

A.2 Host FSM Variables

The interactions between the host and gateway can be described via a simple finite-state machine, since the protocol is simple and the gateway does not maintain any state information. For this FSM, we assume that the host has the following variables or constants:

GID Associated gateway ID (14 bits).

HID Its unique host ID (16 bits).

READY A Boolean indicating whether the IP layer is ready for communications.

LGL_GWY Network-specific logical gateway address (14 bits).

DFL_GWY Default gateway internetwork address (32 bits).

OUR_ADDR Current internetwork address of the host comprised of GID, the ID of its currently associated gateway, and HID, the host's unique ID (32 bits).

ADDR_TBL Table of currently active addresses. This table should contain the local host IP address used for every TCP connection.

Note the construct "GID|HID" denotes the concatenation of the GID and HID values to form a 32-bit internetwork address as previously discussed; the value of the two high-order IP address bits is always assumed to be "10," denoting this address as a class-B address.

In addition to these variable for the host-gateway protocol, other tables will be required in the host implementation to maintain redirect and readdress information.

A.3 Host Finite State Machine

The following FSM is a minimal description of the host's actions and does not contain states or transitions to handle every possible error condition; rather it is intended only to describe the basic protocol actions.

Power-up: Initialize GID to 0; reset ADDR_TBL to empty; set OUR_ADDR to be GID|HID; sets READY to false; go to state REQUEST.

SEARCH: Set a timer to detect failure to find and associate with a gateway within 60 seconds. Go to state REQUEST.

REQUEST: Send an ICMP ECHO packet to the network-specific logical gateway address LGL_GWY. The ECHO packets have an IP destination address of 0x0000|LGL_GWY and an IP source address of OUR_ADDR. Continue sending the ECHO packets periodically until an ECHO-REPLY is received (then go to state REPLY) or the search timer expires (then go to state TIME-OUT).

TIME-OUT: Set OUR_ADDR to 0x0000|HID; set the READY flag to TRUE; and set the default gateway address to 0 (indicating no gateway available). The host should update its NAME-to-ADDRESS entry in the name server by mechanisms not specified by the reconstitution protocol. Go to state REQUEST to resume looking for a gateway.

REPLY: Upon receiving a valid ECHO-REPLY, set GID equal to the gateway ID of the source of the ECHO-REPLY; set OUR_ADDR to GID|HID; set READY flag to TRUE; and set the default gateway address DFL_GWY equal to the IP source address of the ECHO-REPLY. The host should update its NAME-to-ADDRESS entry in its name server by mechanisms not specified by the reconstitution protocol. If ADDR_TBL is not empty, go to state FORWARD; else go to state ECHO.

FORWARD: Upon affiliating with a gateway, send an ICMP FORWARD-REQUEST packet to the default gateway containing the list of currently active addresses for which traffic forwarding should be provided. Continue sending the FORWARD-REQUEST packet periodically until a FORWARD-REPLY packet is received and then go to state ECHO. If a FORWARD-REPLY is not received with a time-out period, declare the gateway down and go to state SEARCH.

ECHO After conducting the gateway acquisition protocol, monitor the state of the acquired gateway by periodically sending ICMP ECHO packets and waiting for the ECHO-REPLYs. If the gateway misses M out of N ECHO-REPLYs, declare the gateway down and go to state SEARCH. Because different types of networks have different rates and dynamics, the up-down parameters are network-specific.

A.4 Gateway Functions

The gateway does not have a FSM corresponding to the host's FSM. However, the gateway does maintain a forwarding database by handling FORWARD-REQUEST and ECHO packets. The actions taken by the gateway when interacting with hosts are described below:

Power-Up: The gateway resets its host forwarding table.

Receive ECHO: When an ICMP ECHO packet is received, it is processed as either a normal ECHO or a reconstitution protocol packet depending on its contents:

- If the IP source address is not Class-B, return an ordinary ECHO-REPLY.
- If the IP source address HID is not equal to the local-network source address, return ECHO-REPLY.
- If the IP source address GID is 0 or the gateway's GID, return ECHO-REPLY and delete any entries for this host from the forwarding table.

Receive FORWARD-REQUEST: When an ICMP FORWARD-REQUEST packet is received, validate it:

- If the IP source address GID equals the gateway's GID (host affiliated with this gateway), a gateway-gateway FORWARD message is reliably sent to each of the previously associated gateways listed in the FORWARD-REQUEST by way of the RP gateway-gateway protocol.
- Otherwise, the packet is ignored.

A.5 Host Traffic Management

The other significant change to the current system from the host's point of view is explicit specification for the traffic management procedures that must be implemented in the host. By traffic management we mean the procedures for selecting the local network destination for a given packet and the procedures for handling faults such as TCP retransmission time-out.

Currently in IP, each implementor is free to implement or not a variety of schemes for selection of local network destinations, handling of gateways, and performing fault isolation. A general set of guidelines for internetwork hosts is specified in [14]. For RP, hosts must always first route packets through the affiliated gateway, which may redirect traffic to another gateway or directly to the destination host.

Most host implementation are not designed to cope with handling network faults and usually depend on error notifications from the underlying network or simply time-out the TCP connections. In the case of the RP experiments, the goal is to avoid disrupting the TCP connections; hence fault isolation mechanisms are defined and implemented in RP hosts so that TCP connections recover after a topology change.

A.6 Network-Specific Details

Although the host-gateway protocol is generally network independent, there are a variety of details, such as logical address assignments, that are specific to each type of network.

A.6.1 PRNET Details

For the PRNET, the 16-bit local network address is directly copied from the host ID field of the reconstitution protocol address.

For the gateway-finding protocol, the network-specific logical address of the gateway is 000B (hex).

A.6.2 ARPANET Details

For the ARPANET, the 24-bit network address is constructed by taking the ARPANET host number for the high 8-bits of the HID and the IMP number from the low 8 bits of the HID. Note that the ARPANET does not currently support logical addressing, so it is not generally possible to disconnect a host from the ARPANET and connect it to another network and have the TCP connections survive uninterrupted.

Until the ARPANET has logical addressing implemented, the hosts and gateways must contain a table of all potential gateways.

A.7 Unique Host ID Assignment

The reconstitution protocols require that identifiers be assigned to each host and gateway-half and that these identifiers are globally unique. While the gateways could perform a translation between unique host ID and physical address, for simplicity in the experiments we utilized a scheme where the host ID and physical address were identical.

PRNET host addresses have been assigned in the range from 1001 (hex) to 1099 (hex) and from 2001 (hex) to 2099 (hex), although in theory the address range is from 1001 (hex) to 7FFF (hex). ARPANET/MILNET addresses have been assigned in the range 0001 (hex) to 07FF (hex). Thus the natural host address assignment for PRNET and ARPANET hosts is nonoverlapping and can be used as-is for the unique host ID values.

For PRNET hosts, the host ID is normally the same as the equipment serial number. For PRNET-to-PRNET gateways, this poses a problem and we assign one gateway-half an address of 10xx and the other side an address of 20xx, etc., again making sure there is not conflict with other PRNET and ARPANET host IDs.

Because the ARPANET does not support logical addressing, its host IDs must be based on the physical network address, thus limiting the ability of a host to migrate from ARPANET to PRNET in these experiments. The use of network addresses for unique IDs also poses a problem for networks with overlapping address ranges, such as ARPANET and MILNET; a translation step between unique host IDs and physical address must be added to an operational RP system.

A.8 New IP Options

Two new options have been defined on an experimental basis for use in the reconstitution experiment. These IP options provide a mechanism for transporting the TCP connection identifier (original host address) in packets where the source or destination hosts have changed addresses. Neither, one, or both of these options may be present in an IP packet, in addition to any other IP options.

A.8.1 Old-Source-Address Option

The old-source-address option is used to carry the original source address in a packet after an address change. Upon receipt at the destination host, the address in this option is copied into the source address field of the IP header (or pseudo-header) before passing the packet up to TCP.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |  Type = 138   |    Len = 6    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Old Source Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.8.2 Old-Destination-Address Option

The old-destination-address option is used to carry the original destination address in a packet after an address change. Upon receipt at the destination host, the address in this option is copied into the destination address field of the IP header (or pseudo-header) before passing the packet up to TCP.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |  Type = 139   |    Len = 6    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Old Destination Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.9 Host-Gateway Packet Formats

The RP protocol has also defined extensions to the ICMP protocol that operates between hosts and gateways. These extensions include a change in meaning and usage of the ECHO/ECHO-REPLY packets and three new packet types to control the forwarding of data packets after an address change.

For all of the following packet definitions, the IP header is constructed normally with a protocol value of 1 (ICMP). The ICMP header contains the normal type/code and checksum fields.

The packet definitions are formatted in standard network-order: least-significant byte of multibyte fields first.

ECHO

The ICMP ECHO packet is used for two purposes: In the normal internetwork system, the ECHO packet is used to determine if the destination host is operating. For the reconstitution protocol, the ECHO packet has the additional function of requesting an association between host and gateway.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 8    |   Code = 0    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Identifier           |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Optional Echo Data                       |
.                                                               .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Local Network fields:

Destination ID Network-specific gateway logical address.

Source ID Network-specific host source address.

IP Header fields:

Destination Address Destination gateway address formed by setting the GID to 0 and using the network-specific logical gateway address as the HID:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|             0             |0 0|   Logical Gateway ID      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source Address Current IP address of host.

Protocol 1 for ICMP protocol.

ICMP fields:

Type 8 for ECHO messages; 0 for ECHO-REPLY messages.

Code Always 0.

Identifier Copied into the ECHO-REPLY; identifier may be used by the host to match ECHO messages with ECHO-REPLYs.

Sequence Number Copied into the ECHO-REPLY; sequence number may be used by the host.

Optional Data The data received in the ECHO is returned in the ECHO-REPLY.

ECHO REPLY

The ICMP ECHO-REPLY packet is sent in response to an ECHO packet.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0    |   Code = 0    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Identifier           |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Optional Returned Echo Data                  |
.                                                               .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Header fields:

Destination Address IP address of host that sent the ECHO packet.

Source Address IP address of gateway.

Protocol 1 for ICMP protocol.

ICMP fields:

Identifier Copied from the ECHO message; identifier may be used by the host to match ECHO messages with ECHO-REPLYs.

Sequence Number Copied from the ECHO message; sequence number may be used by the host.

Optional Data Any data received in the ECHO is returned in the ECHO-REPLY.

• FORWARD-REQUEST

The ICMP FORWARD-REQUEST packet is sent to a gateway by a host to request forwarding for a set of previous addresses that are still in use.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 18   |   Code = 0    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Identifier           |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Old Host Address #1                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Old Host Address #n                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Header fields:

Destination Address Destination gateway address formed by addressing the gateway as a host associated with itself. That is:

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|   Unique Gateway ID       |0 0|  Unique Gateway ID        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Source Address: Current IP address of host.

ICMP fields:

Identifier: Copied into the FORWARD-REPLY, identifier may be used by the host to match FORWARD-REQUEST messages with FORWARD-REPLYs.

Sequence Number: Copied into the FORWARD-REPLY, sequence number may be used by the host.

Old Host Addresses: Copied into the FORWARD-REPLY, the data field contains a list of old IP addresses for which forwarding service is requested. Normally these address will be associated with currently open TCP connections.

FORWARD-REPLY

The ICMP FORWARD-REPLY packet is sent to the host from the gateway in response to receiving a valid FORWARD-REQUEST packet.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 20   |   Code = 0    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Identifier           |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Old Host Address #1                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Old Host Address #n                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Header fields:

Destination Address: IP address of the host.

Source Address: IP address of the gateway.

ICMP fields:

Identifier: Copied from the FORWARD-REQUEST; identifier may be used by the host to match FORWARD-REQUEST messages with FORWARD-REPLYs.

Sequence Number: Copied from the FORWARD-REQUEST; sequence number may be used by the host.

Old Host Addresses: Copied from the FORWARD-REQUEST; the data field contains a list of old IP addresses for which forwarding service is requested.

READDRESS

The ICMP READDRESS message is used to notify a host of a change in address for one of the destinations currently active.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type = 17     | Code = 0      |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       New Host Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Internetwork Header + 64 bits of Original Datagram Data         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IP Header fields:

Destination Address: IP address of the host.

Source Address: IP address of the sending gateway.

ICMP fields:

New Host Address: The current address for the host indicated in the destination address field of the enclosed IP header.

Enclosed IP Header: The enclosed IP header plus 64-bits of the datagram data field is copied from the packet that triggered sending of the READDRESS; the enclosed IP header can be used to fully identify the destination host, protocol, and connection, if necessary.

Appendix B

IPLI Compatibility Considerations

Within the Strategic C^3 Experiment, it is desirable that the IPLI security devices currently under development and the reconstitution gateways interoperate to provide a survivable, reconstitutable, and secure communications system.

The original reconstitution protocol design placed a burden on the host as well as on the gateway to implement new protocol features. In the case of the IPLI, we are limited in the number and breadth of changes that can be made to the device for security, configuration control, and development time and cost reasons. In addition, the changes for the reconstitution protocol should use existing IP protocol features and must be backward-compatible with existing gateways.

The new features of the reconstitution protocols of concern are:

- All addresses are of class-B type.
- Hosts have permanent, unique host identifiers that are independent of the current local network device address.
- Hosts execute a gateway acquisition algorithm that lets them find a working gateway on the network, specify unique host ID to local network device ID mapping, get a new associated-gateway address to use in the class-B network field, and specify all previous addresses that are currently in use.
- Gateways add an IP option to readdress destination hosts.
- Hosts add an IP option to readdress themselves.

In the case of the IPLI, we are very limited in the number and extent of changes that can be made to the network code. Thus we need to find a way to satisfy the new features of the reconstitution protocol using existing IP mechanisms.

The following problem areas and proposed solutions demonstrate how additional mechanisms could be implemented in the gateway to shield the host from knowledge of the operation of RP. Some of the suggestions were actually incorporated into the RP protocol and gateway implementation (such as using ECHO/ECHO-REPLY for host affiliation); others (such as the gateway removing OLD ADDRESS options) were not, since the IPLIs were not available for use during the experiments.

B.1 Address Format

For the PR-IPLI, the requirement that all addresses be similar to the existing class-B format is no problem, since the PRNET is already a class-B network. For the ARPANET-IPLI, the requirement is a problem since the ARPANET is normally a class-A network.

Solution: Implement a class-B ARPANET address mapping in the IPLI; this can be accomplished with a small amount of code change.

B.2 Unique Host Identifiers

The reconstitution protocol design provides each host with a permanent and unique identifier that is independent of its current local network device address. Since not all networks provide a logical address facility, the translation between unique ID and local network address is a function that is provided by the gateways. This function requires that hosts specify their local network mapping in the gateway acquisition packet, implement host-specific redirects, and always send the first packet to a destination to gateway even if the destination appears to be on the local network.

Solution: Since the IPLI will not execute the gateway acquisition protocol (see below), the translation function between unique host IDs and local network device addresses cannot be provided in the gateway. Thus the unique host ID must directly translate into the local network identifier. There is no loss in functionality for PR-IPLIs since the PRNET supports logical addressing. On the ARPANET, we must restrict ARPANET-IPLIs from being relocated to a new IMP without a change to the unique ID; again, there is no significant loss in functionality for the planned SAC C3 experiments.

B.3 Gateway Acquisition Protocol

The gateway acquisition protocol is executed between a host and its associated gateway. This protocol explicitly provided for:

- Finding a currently operational gateway.
- Specifying the translation between unique host ID and network device ID.
- Explicitly agreeing between a host and gateway that the host is now associated with the indicated gateway.
- Monitoring the up-down link-state of the host and gateway.
- Getting the new associated gateway number to put in the "network" field.
- Specifying previous addresses that are still in use.

Implementation of the originally specified gateway acquisition protocol would be a substantial change to the IPLI network code. Thus we propose that the IPLI not implement this protocol, but that these functions will be handled as specified below.

B.3.1 Finding Operational Gateways

One of the major deficiencies of the current ARPANET and PRNET protocols is lack of a dynamic mechanism for finding operational gateways; each host maintains a static table of possible gateway addresses to try. The reconstitution protocol instead defines a dynamic mechanism based around logical addressing.

Solution: No change is needed in the IPLI. The reconstitution gateways will return a redirect for any packet addressed to the PRNET gateway logical address "B," not just gateway acquisition packets. The ARPANET-IPLI will need a table with all possible reconstitution gateway addresses, as is currently required.

B.3.2 Monitoring Host-Gateway Status

The gateway acquisition protocol provides a mechanism whereby the host and the gateway can monitor the status of the other. This monitoring allows the gateway to filter old or unappropriate forward-request packets and to return destination-unreachable ICMP messages when the host is declared dead.

Solution: *The IPLI will periodically send ICMP echo packets to its associated* (or prime) gateway and will respond to ICMP echo packets from a gateway. Thus ICMP probes will be used to check status, and the gateway will return destination unreachable only upon indication from the local network.

B.3.3 Getting the New Associated Gateway Number

When a host moves, it should change its internetwork address to reflect its new location to reduce network overhead traffic. The new associated gateway number is specified in the gateway acquisition message.

Solution: The reconstitution gateway will place its address in ICMP echo-reply packets; a host can thus obtain its new address. However, to reduce changes to the IPLI, the IPLI need not implement the address changing mechanism but rather will keep its same address for the duration of each experiment. While this will introduce additional network overhead traffic, the added overhead will not be excessive.

B.3.4 Specifying Previous Addresses

When a host "moves" or changes its association to a new gateway, it must explicitly tell its new gateway the previous addresses that are still in use. This information allows the new gateway to send forward requests to the previously associated gateways asking that traffic be forwarded to the new address. The IPLI has only a single, unchanging address that it uses to exchange up-down information with its peer IPLIs.

Solution: The reconstitution gateways will take the source address from any ICMP echo packet as being one of the previous addresses that is still in use. No change to the IPLI, as long as it generates ICMP echo packets

B.4 Handling of Old-Destination-Address IP Option

When gateway forwards a packet for a host that has moved, it puts the original IP address in the packet as an IP option and places the new destination address in the IP header. Thus the IP header destination field always specifies the destination of the packet.

The forwarding gateway also sends an ICMP readdress packet to the source, so that the source can directly address the destination and eliminate the gateway forwarding function.

Solution: To avoid change to the IPLI, the last gateway will remove the original IP destination address from the options field and place it in the destination field of the IP header before sending the packet to the host. Thus the IPLI will be unaware of its change in address. The gateway will also not generate readdress ICMP messages.

B.5 Handling of Old-Source-Address IP Option

Normally, when a host moves, it places its original IP address in the packet as an IP readdress-source option and uses its new address in the IP header. This allows the destination of the packet to return information direct to the source using the new address.

Solution: The IPLI will not implement the readdress-source IP option since the destination IPLI will not implement the readdress-destination IP option. Thus no changes are needed to the IPLI.

B.6 More Sophisticated Fault-Isolation Mechanisms

To utilise fully the survivability provided by the reconstitution protocol gateways, hosts need to provide more sophisticated fault-isolation mechanism that are currently used. Since fault isolation features can be generic to the internetwork or specific to a given network, three cases are described separately.

ARPANET Fault Isolation—The ARPANET-IPLI must respond to ARPANET destination-unreachable messages. If sending normal data traffic, the host-specific redirect entry should be erased since it is no longer valid, and the packet should be sent to the IPLI's associated gateway for forwarding. If the IPLI is echoing off a gateway, the destination-dead message indicates that gateway is down and the IPLI should select another gateway. While these mechanisms are needed for the normal ARPANET-IPLI, their functioning is more critical for the reconstitution system.

PRNET Fault Isolation—PRNET-IPLI fault isolation is more complex than that of ARPANET-IPLI because the PRNET does not provide failure information as complete as that of the ARPANET. The IPLI must respond to unreachable flags in the PRNET device advisory packet (DAP). If the IPLI is sending normal data traffic, the host-specific redirect entry should be erased since it is no longer valid and the packet should be sent to the IPLI's associated gateway for forwarding. If the IPLI is echoing off a gateway, the unreachable flag indicates that gateway is down and the IPLI should select another gateway. Again, these mechanisms are needed for the normal PRNET-IPLI but are very important in the reconstitution system.

Internetwork Fault Isolation—In the reconstitution protocols, each host is responsible for maintaining an association with a gateway that can be directly reached through the local network. Thus the time the system takes to respond to a network partition or to a host moving is directly related to the time that the host takes to change its gateway association. Thus the IPLI must periodically echo its associated gateway to check for changes in topology.

B.7 Initialization

The IPLI needs to get an initial network number (associated gateway ID). Normally, the host would obtain this initial value from its first acquired gateway.

Solution: Since the IPLI doesn't implement this protocol, the initial value will need to be manually entered into the IPLI during initialization. Since the IPLI needs to be manually initialized anyway, this is no problem. The initial value must correspond to an actual gateway ID that is on the network attached to the IPLI.

B.8 Summary of Recommendations

By limiting the functionality of the reconstitution protocol, taking advantage of the simple host characteristics of the IPLI, and changing the gateway functions slightly, we can interoperate with the IPLI with no major changes to the IPLI network code.

The changes or functionality needed are:

- Class-B ARPANET address mapping.
- Periodically send ICMP Echo packets to prime (or associated) gateway.
- Answer ICMP Echo packets with Echo-reply.
- Host-specific redirect processing.
- Good fault isolation procedures.

All of these features are either backward-compatible with the existing gateways, or should be in a good host implementation, or are useful for future applications of the IPLI.

Appendix C

Gateway-Gateway Protocol

The gateway to gateway protocol used in the RP experiments is modeled after the GGP used in the LSI-11 ARPANET gateways. While GGP is being superceded by an SPF-type of routing protocol in the new butterfly gateways, it is fairly well documented and considered a stable protocol.

For all of the following packet definitions, the IP header is constructed normally with a protocol value of 9 (IGP). IGP is a generic identification of any internal gateway-gateway protocol, of which, GGP is one instance.

The checksum in the header is the standard 1's complement sum. It is computed as the 16-bit one's complement of the one's complement sum of the IGP message starting with the Type field. For computing the checksum, the checksum field should be sero.

The packet definitions are formatted in standard network-order: least-significant byte of multibyte fields first.

ROUTING UPDATE

The IGP routing update packets contain a full routing table and is sent periodically to all cross-net and common-gateway neighbors. Routing updates are not acknowledged.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 16   |   N-groups    |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Unused = 0   |  Distance #1  |    Marking    | # gwy-halves  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway-Half Address (1,1)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway-Half Address (1,2)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway-Half Address (1,j)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Unused = 0   |  Distance #i  |    Marking    | # gwy-halves  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway-Half Address (i,1)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway-Half Address (i,2)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway-Half Address (i,j)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IGP fields:

N-groups: The number of update groups in this packet.

Sequence Number: The 16-bit sequence number used the most recent update.

Length: The length of the packet in bytes.

Distance: An 8-bit hop count which applies to each group of gateway-halves at the same distance.

Marking: This 8-bit field represents the marking of the gateway-halves in this distance group. If there are different markings for the same distance group, they are considered as different distance groups.

# Gwy-halves: The number of gateway-halves which are reported in this distance group.

Gwy-half Address (i,j) The 16-bit gateway ID left-justified in a 32-bit internetwork address field. The address (i, j) is for the j-th gateway in distance group i.

Gateway-to-Gateway Forward Request

The gateway-to-gateway forward request message is transmitted from one gateway to all its cross-net neighbors informing them of host forwarding information. It is transmitted for 3 times at an interval of 60 seconds to increase the probability of reaching the destination.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 19    |   N-groups    |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Old Host Address #1                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     New Host Address #1                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Old Host Address #n                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     New Host Address #n                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IGP fields:

N-groups: The number of host address-pair groups in this packet.

Sequence Number: The 16-bit sequence number used the most recent update.

Length: The length of the packet in bytes.

Old Address #n: The old address of the host #n.

New Address #n: The corresponding new address for host #n.

GATEWAY ECHO REQUEST

The gateway echo request is used as a "ping" packet from one gateway to another gateway to check to see if that neighbor is still alive.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 17   |  Unused = 0   |          Unused = 0           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IGP fields:

Length: The length of the packet in bytes.

GATEWAY ECHO REPLY

The gateway echo reply packet is used to answer the echo request packet.

```
                    1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 18   |  Unused = 0   |          Unused = 0           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IGP fields:

Length: The length of the packet in bytes.

Bibliography

[1] Vinton G. Cerf and Edward Cain. The DoD internet architecture model. *Computer Networks*, 7(5):307–318, October 1983.

[2] Vinton G. Cerf. *Internet Addressing and Naming in a Tactical Environment.* IEN 110, Information Processing Techniques Office, Defense Advanced Research Projects Agency, Arlington, Virginia, August 1979.

[3] Carl A. Sunshine. Addressing problems in multi-network systems. In *Proceedings of IEEE INFOCOM 82*, pages 311–317, March-April 1982.

[4] Robert E. Kahn, Steven A. Gronormeyer, Ronald C. Kunzelman, and Jerome Birchfeld. Advances in packet radio technology. In *Proceedings of the IEEE*, pages 1468–1496, November 1978.

[5] Michael Frankel, Bob Baker, Emilie Siarkiewicz, and Robert Kahn. *Strategic Command, Control, and Communications Experiment: Achievements and Plans.* Technical Report, Defense Advanced Research Projects Agency, Strategic Air Command, Defense Communications Agency, December 1984.

[6] Radia Perlman. *Flying Packet Radios and Network Partitions.* IEN 146, Bolt, Beranek and Newman, June 1980.

[7] Jonathan B. Postel and Carl A. Sunshine. *Addressing Mobile Hosts in the ARPA Internet Environment.* IEN 135, Information Sciences Institute, University of Southern California, Marina del Rey, California, March 1980.

[8] Zaw-Sing Su and James E. Mathis. *Internetwork Accommodation of Network Dynamics: Reconstitution Protocol.* Interim Technical Report, SRI International, February 1986.

[9] Zaw-Sing Su and James E. Mathis. Internetwork accommodation of network dynamics: naming and addressing. In *Proceedings of 7th International Conference on Computer Communications*, pages 681–685, November 1984.

[10] Zaw-Sing Su and James E. Mathis. Internetwork accommodation of network dynamics: organizational structure. In *Proceedings of IEEE INFOCOM 85*, pages 428–430, March 1985.

[11] Rodney J. Reining, Dwight F. Hare, and James E. Mathis. *F5/10 Demonstration.* Special Demonstration Report, SRI International, November 1986.

[12] Rodney J. Reining, Dwight F. Hare, and James E. Mathis. *F8/11 Demonstration.* Special Demonstration Report, SRI International, November 1986.

[13] Rodney J. Reining and Dwight F. Hare. *F9 Demonstration.* Special Demonstration Report, SRI International, January 1987.

[14] David Clark. *Fault Isolation and Recovery.* RFC 816, MIT Laboratory for Computer Science, July 1982.

DISTRIBUTION LIST

DeFranco RADC/COTD	10
RADC/DOVL GRIFFISS AFB NY 13441	1
RADC/DAP GRIFFISS AFB NY 13441	2
ADMINISTRATOR DEF TECH INF CTR ATTN: DTIC-DDA CAMERON STA BG 5 ALEXANDRIA VA 22304-6145	12
RADC/COTD BLDG 3, ROOM 16 GRIFFISS AFB NY 13441-5700	1
AFCSA/SAMI WASHINGTON DC 20330-5425	1
HQ USAF/SCTT WASHINGTON DC 20330	1
HQ USAF/RDSS WASHINGTON DC 20330-5040	1
OASD (C3I), INFORMATION SYSTEMS ROOM 3E187 WASHINGTON DC 20301-3040	2
HQ AFSC/DLAE ANDREWS AFB DC 20334-5000	1

HQ AFSC/SDE 1
ANDREWS AFB DC 20334-5000

HQ AFSC/XRTD 1
ANDREWS AFB DC 20334

HQ SAC/SlPT 1
OFFUTT AFB NE 68113-5001
EOP

HQ ESC/DOOA 1
SAN ANTONIO TX 78243-5000

HQ AFOTEC (OAWD) 1
Attn: Capt. Novack)
KIRTLAND AFB NM 87117-7001

AFHRL/OTS 1
WILLIAMS AFB AZ 85240-6457

HQ SPACECOM/XPYX 1
EOP
ATTN: DR. WILLIAM R. MATOUSH
PETERSON AFB CO 80914-5001

CODE H396RL TECHNICAL LIBRARY 1
DEFENSE COMMUNICATIONS
ENGINEERING CENTER
1860 WIEHLE AVENUE
RESTON VA 22090

COMMAND CONTROL AND COMMUNICATIONS DIV 1
DEVELOPMENT CENTER
MARINE CORPS DEVELOPMENT & EDUCATION COMMAND
ATTN: CODE DIOA
QUANTICO VA 22134

COMMANDER 1
NAVAL OCEAN SYSTEMS CENTER
ATTM: TECHNICAL LIBRARY, CODE 9642
SAN DIEGO CA 92152-5000

NAVELEXSYCOM 1
PDE-110-33
WASHINGTON DC 20363

Advisory Group on Electron Devices 2
201 Varick Street, Suite 1140
New York NY 10014-4877

HQ, Fort Huachuca 1
TECH REF DIV
ATTN: BESSIE BRADFORD
Ft. Huachuca AZ 85613-6000

HQ ESD/XRX 1
HANSCOM AFB MA 01731

ESD/XRC (AFSC) 1
HANSCOM AFB MA 01731-5000

SRI International 5
333 Ravenswood Ave
Menlo Park CA 94025-3493

Defense Advanced Research projects Agency 4
1400 Wilson Blvd
Arlington VA 22209
EOP
Attn: Dr. Dennis Perry

Computer Corporation of America 2
4 Cambridge Center
Cambridge MA 02139
Attn: D. Smith

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. The areas of technical competence include communications, command and control, battle management, information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic, maintainability, and compatibility.